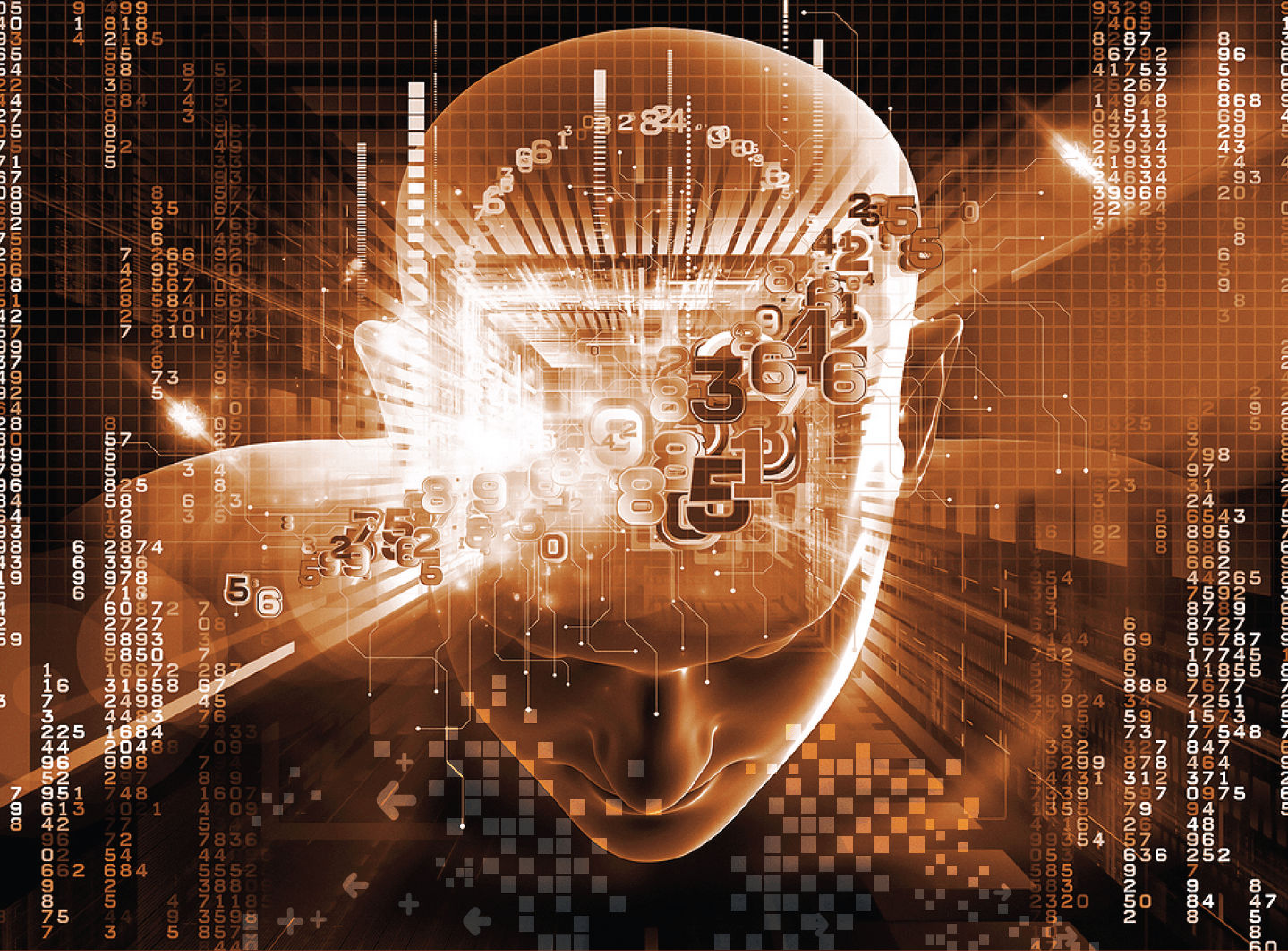


模式识别与人工智能

（基于MATLAB）

周润景◎编著

10余年模式识别领域教学与科研成果总结，理论知识系统全面，案例实践性强，所用数据为酒瓶颜色的RGB三基色原始数据，数据量大，信息丰富

清华大学出版社

模式识别与人工智能

（基于 MATLAB）

周润景　编著

清华大学出版社
北　京

内容简介

本书将模式识别与人工智能理论和实际应用相结合，以酒瓶颜色分类为例，介绍各种算法理论及相应的 MATLAB 实现程序。全书共分为 10 章，内容包括模式识别概述、贝叶斯分类器设计、判别函数分类器设计、聚类分析、模糊聚类分析、神经网络聚类设计、模拟退火算法聚类设计、遗传算法聚类设计、蚁群算法聚类设计、粒子群算法聚类设计，覆盖了各种常用的模式识别技术。

本书可作为高等院校自动化、计算机、电子和通信等专业研究生和高年级本科生的教材，也可作为计算机信息处理、自动控制等相关领域工程技术人员的参考用书。

图书在版编目(CIP)数据

模式识别与人工智能：基于 MATLAB/周润景编著. —北京：清华大学出版社，2018
ISBN 978-7-302-48635-0

Ⅰ. ①模…　Ⅱ. ①周…　Ⅲ. ①模式识别－MATLAB 软件 ②智能计算机－MATLAB 软件
Ⅳ. ①O235 ②TP387

中国版本图书馆 CIP 数据核字(2017)第 261766 号

责任编辑：袁金敏
封面设计：刘新新
责任校对：李建庄
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：25　　**字　　数**：610 千字
版　　次：2018 年 7 月第 1 版　　**印　　次**：2018 年 7 月第 1 次印刷
印　　数：1～2500
定　　价：89.00 元

产品编号：076802-01

前言

FOREWORD

随着模式识别技术的迅猛发展，目前该技术已经成为当代高科技研究的重要领域之一，不仅取得了丰富的理论成果，而且其应用范围越来越广泛，几乎遍及各个学科领域，如人工智能、机器人、系统控制、遥感数据分析、生物医学工程、军事目标识别等。由于其在国民经济、国防建设、社会发展的各个方面得到了广泛应用，因而越来越多的人认识到模式识别技术的重要性。

本书以实用性为宗旨，以对酒瓶颜色的分类设计为主，将理论与实践相结合，介绍了各种相关分类器设计。

第 1 章介绍模式识别的概念、模式识别的方法及其应用。

第 2 章讨论贝叶斯分类器的设计。首先介绍贝叶斯决策的概念，让读者对贝叶斯理论有所了解，然后介绍基于最小错误率和最小风险的贝叶斯分类器的设计，将理论应用到实践，让读者真正学会运用该算法解决实际问题。

第 3 章讨论判别函数分类器的设计。判别函数包括线性判别函数和非线性判别函数，本章首先介绍判别函数的相关概念，然后介绍线性判别函数 LMSE 和 Fisher 分类器的设计及非线性判别函数 SVM 分类器的设计。

第 4 章讨论聚类分析。聚类分析作为最基础的分类方法，涵盖了大量经典的聚类算法及衍生出来的改进算法。本章首先介绍相关理论知识，然后依次介绍 K 均值聚类、K 均值改进算法、KNN 聚类、PAM 聚类、层次聚类及 ISODATA 分类器设计。

第 5 章讨论模糊聚类分析。首先介绍模糊逻辑的发展、模糊数学理论、模糊逻辑与模糊推理等一整套模糊控制理论，然后介绍模糊分类器、模糊 C 均值分类器、模糊 ISODATA 分类器及模糊神经网络分类器的设计。

第 6 章讨论神经网络聚类设计。首先介绍神经网络的概念及其模型等理论知识，然后介绍基于 BP 网络、Hopfield 网络、RBF 网络、GRNN、小波神经网络、自组织竞争网络、SOM 网络、LVQ 网络、PNN、CPN 的分类器设计。

第 7 章讨论模拟退火算法聚类设计。首先介绍模拟退火算法的基本原理、基本过程，然后介绍其分类器的设计。

第 8 章介绍遗传算法聚类设计，包括遗传算法原理及遗传算法分类器设计的

详细过程。

第9章介绍蚁群算法聚类设计，包括蚁群算法的基本原理、基于蚁群基本算法的分类器设计和改进的蚁群算法 MMAS 的分类器设计。

第10章介绍粒子群算法聚类设计，包括粒子群算法的运算过程、进化模型、原理及其模式分类的设计过程。

本书没有像大多数模式识别的书那样讲解烦琐的理论，而是简明扼要地介绍每一种算法的核心，并通过大量的实例介绍模式识别知识。书中针对每一种模式识别算法，按理论基础和实例操作两部分进行介绍。在读者掌握基础理论后，通过实例可以了解算法的实现思路和方法；进一步掌握核心代码编写，就可以很快掌握模式识别技术。

本书内容来自作者的科研与教学实践。读者在学会各种理论和方法后，可将书中的不同算法加以改造应用于自己的实际工作。

本书第1～3章由李楠编写，其余由周润景完成并统稿、定稿。参加本书编写的还有邵盟、南志贤、刘波、李艳、邵绪晨、冯震、崔婧、任自鑫、谢亚楠、祖晓玮、张赫、丁岩、井探亮、邢婧、陈萌。

在本书的编写过程中，作者力求完美，但由于水平有限，书中难免有不足之处，敬请指正。

作　者

2018年3月

目录
CONTENTS

第1章

模式识别概述

1.1 模式识别的基本概念

模式识别(pattern recognition)也称机器识别,就是通过计算机用数学技术方法来进行模式的自动处理和判读。该学科的主要任务是利用计算机进行模拟人的识别能力,提出识别具体客体的基本理论与实用技术。其最基本的方法是计算,即计算要识别的事物与已知的标准事物的相似程度,从而让机器能判别事物。因此,找到度量不同事物差异的有效方法是研究的关键。随着计算机技术的发展,人类有可能研究复杂的信息处理过程。我们把环境与客体统称为"模式"。信息处理过程的一个重要形式是生命体对环境及客体的识别。对人类来说,特别重要的是对光学信息(通过视觉器官来获得)和声学信息(通过听觉器官来获得)的识别。这是模式识别的两个重要方面。模式识别是确定一个样本的类别属性的过程,即把某一样本归属于多个类型中的某个类型。例如,数据分类,结果就是将待分类数据按属性分类;指纹识别,就是使用指纹的总体特征如纹形、三角点等来进行分类,再用局部特征如位置和方向等来进行识别用户身份;还有语音识别、生物认证、字符识别等。

1.1.1 模式的描述方法

对于模式的描述方法有两种:定量描述和结构性描述。其中,定量描述通过对事物的属性进行度量,用一组数据来描述模式;结构性描述是对事物所包含的成分进行分析,用一组基元来描述模式。

针对定量描述方法,一个具体的研究对象称为样品。对于一个样品来说,必须确定一些与识别有关的因素作为研究的根据,每一个因素称为一个特征。模式用样品的一组数据来描述。模式的特征集一般可以用特征向量表示。特征向量中的每个元素称作特征。假设一个样品 $\boldsymbol{X}$ 有 n 个特征,若用小写字母 x 来表示特征,则可以把 $\boldsymbol{X}$ 看作一个 n 维列向量,该向量 $\boldsymbol{X}$ 称为特征向量,记作:

$$X = (x_1, x_2, \cdots, x_n)^T \tag{1-1}$$

模式识别问题就是根据 $\boldsymbol{X}$ 的 n 个特征来判别模式 $\boldsymbol{X}$ 属于 $\omega_1, \omega_2, \cdots, \omega_M$ 类中的哪一类。其目的是在特征空间和解释空间之间建立一种特殊的对应关系。其中，特征空间由特征向量构成，包括模式的度量、属性等，通过对具体对象进行观测得到；解释空间由所属模式类别的集合构成。

1.1.2 模式识别系统

模式识别的关键是如何利用计算机进行模式识别，并对样本进行分类。执行模式识别的基于计算机的系统（可以是各种有计算能力的处理器系统）称为模式识别系统。该系统具有两种工作方式，训练方式和识别方式。

图 1-1 是一个典型的模式识别系统，由数据获取、预处理、特征提取、分类决策及分类器设计等部分组成。该系统各个组成部分的功能概括如下。

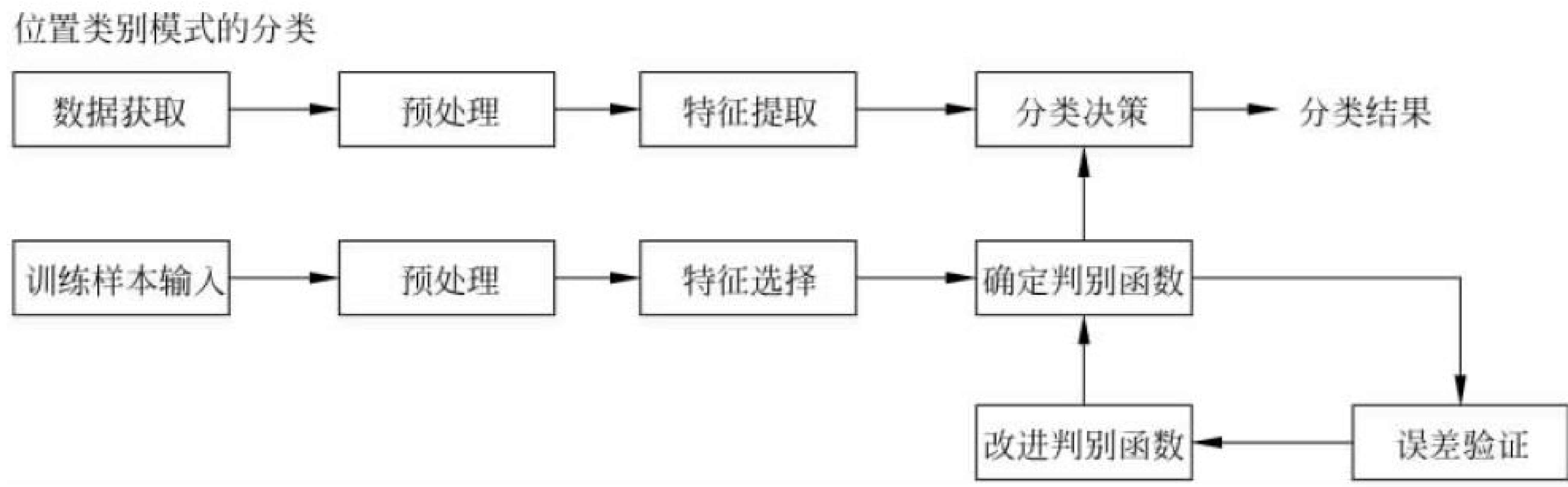


图 1-1　模式识别系统

（1）数据获取：一般情况下，获取的信息类型有以下几种。

- 一维波形：心电图、脑电波、声波、震动波形等。
- 二维图像：文字、地图、照片等。
- 物理参量：体温、化验数据、温度、压力、电流、电压等。

（2）预处理：对由于信息获取装置或其他因素所造成的信息退化现象进行复原、去噪，加强有用信息。

（3）特征提取：由信息获取部分获得的原始信息，其数据量一般相当大。为了有效地实现分类识别，应对经过预处理的信息进行选择或变换，得到最能反映分类本质的特征，构成特征向量。其目的是将维数较高的模式空间转换为维数较低的特征空间。

（4）分类决策：在特征空间中用模式识别方法（由分类器设计确定的分类判别规则）对待识别模式进行分类判别，将其归为某一类别，输出分类结果。这一过程对应于特征空间向类别空间的转换。

（5）分类器设计：为了把待识别模式分配到各自的模式类中，必须设计出一套分类判别规则。基本做法是收集一定数量的样本作为训练集，在此基础上确定判别函数、改进判别函数和误差检验。

其中分类器设计在训练方式下完成，利用样本进行训练，确定分类器的具体参数，而分类决策在识别方式下起作用，对待识别的样本进行分类决策。

1.2 模式识别的基本方法

模式识别作为人工智能的一个重要应用领域，目前得到了飞速发展。针对不同的对象和不同的目的，可以使用不同的模式识别理论或方法。目前，基本的技术方法有如下四种。

1. 统计模式识别

统计模式识别是首先根据待识别对象所包含的原始数据信息，从中提取出若干能够反映该类对象某方面性质的相应特征参数，并根据识别的实际需要从中选择一些参数的组合作为一个特征向量，根据某种相似性测度，设计一个能够对该向量组表示的模式进行区分的分类器，就可把特征向量相似的对象分为一类。

2. 结构模式识别

当需要对待识别对象的各部分之间的联系进行精确识别时，就需要使用结构模式识别方法。结构模式识别是根据识别对象的结构特征，将复杂的模式结构先通过分解划分为多个相对更简单且更容易区分的子模式，若得到的子模式仍有识别难度，则继续对其进行分解，直到最终得到的子模式具有容易表示且容易识别的结构为止，通过这些子模式就可以复原原先比较复杂的模式结构。

3. 模糊模式识别

模糊集理论认为，模糊集合中的一个元素，可以不是百分之百地确定属于该集合，而是可以以一定的比例属于该集合，不像传统集合理论中某元素要么属于要么不属于该集合的定义方式，更符合现实当中许多模糊的实际问题，描述起来更加简单合理。在用机器模拟人类智能时模糊数学能更好地描述现实当中具有模糊性的问题，进而更好地进行处理。模糊模式识别就以模糊集理论为基础，根据一定的判定要求建立合适的隶属度函数来对识别对象进行分类。

正是因为模糊模式识别能够很好地解决现实当中许多具有模糊性的概念，使其成为一种重要的模式识别方法。在进行模糊识别时，也需要建立一个识别系统，需要对实际的识别对象的特征参数按照一定的比例进行分类，这些比例往往是根据人为的经验作为参考值，只要符合认可的经验认识就行，之后建立能够处理模糊性问题的分类器对不同类别的特征向量进行判别。

4. 人工神经网络

神经网络分为四种类型，即向前型、反馈型、随机型和自组织竞争。神经网络作为模式识别技术当中最重要的方法之一，相对于传统的模式识别方法，它具有如下优势。

(1) 神经网络属于学习及自适应能力很强的方法。

(2) 对于任意给定的函数,神经网络都能够无限逼近,这是因为在分类的整个过程中,神经网络通过调整权值不断地明确分类所依据的精确关系。

(3) 神经网络属于非线性模型,这使得它能够灵活地模拟现实世界中的数据之间的复杂关系模型。

1.3 模式识别的应用

模式识别是一种智能的活动,它包括分析和判断两个过程。随着计算机性能的提高、互联网技术的迅速发展、理论研究的深入以及和其他研究领域(如机器学习、数学、统计学、生物学等)的促进、融合,现在模式识别技术已经被普遍地应用于生物学(自动细胞学、染色体特性研究、遗传研究)、天文学(天文望远镜图像分析、自动光谱学)、经济学(股票交易预测、企业行为分析)、医学(心电图分析、脑电图分析、医学图像分析)、工程(产品缺陷检测、特征识别、语音识别、自动导航系统、污染分析)、军事(航空摄像分析、雷达和声呐信号检测和分类、自动目标识别)、安全(指纹识别、人脸识别、监视和报警系统)等领域。

本书基于 MATLAB 实现不同模式识别方法的应用,以酒瓶颜色的分类为主。由不同材料制成的不同颜色和项目的玻璃必须被分类,以获得高质量的可回收原料。在玻璃回收厂,玻璃瓶被分类放置到容器中,然后一起熔化处理产生新的玻璃。在这个过程中,玻璃瓶被分选到不同容器这一步是很重要的,因为生产玻璃的不同客户需要不同的颜色,并且回收的瓶子颜色混杂,对瓶子的分类没有预先设定。在这种情况下,多数操作员根据经验手工分选回收的瓶子,以使生产的玻璃达到期望的颜色。这既费时又费力,而通过模式识别的方法分类,则既解放了生产力,又提高了效率。表 1-1 为 59 组三元色数据,其中前 29 组作为训练数据,后 30 组作为测试数据。前 29 组数据的类别已经给出。

表 1-1 三元色数据

序　　号	A	B	C	所属类别
1	1739.94	1675.15	2395.96	3
2	373.3	3087.05	2429.47	4
3	1756.77	1652	1514.98	3
4	864.45	1647.31	2665.9	1
5	222.85	3059.54	2002.33	4
6	877.88	2031.66	3071.18	1
7	1803.58	1583.12	2163.05	3
8	2352.12	2557.04	1411.53	2
9	401.3	3259.94	2150.98	4
10	363.34	3477.95	2462.86	4
11	1571.17	1731.04	1735.33	3
12	104.8	3389.83	2421.83	4
13	499.85	3305.75	2196.22	4
14	2297.28	3340.14	535.62	2
15	2092.62	3177.21	584.32	2

续表

序 号	A	B	C	所属类别
16	1418.79	1775.89	2772.9	1
17	1845.59	1918.81	2226.49	3
18	2205.36	3243.74	1202.69	2
19	2949.16	3244.44	662.42	2
20	1692.62	1867.5	2108.97	3
21	1680.67	1575.78	1725.1	3
22	2802.88	3017.11	1984.98	2
23	172.78	3084.49	2328.65	4
24	2063.54	3199.76	1257.21	2
25	1449.58	1641.58	3405.12	1
26	1651.52	1713.28	1570.38	3
27	341.59	3076.62	2438.63	4
28	291.02	3095.68	2088.95	4
29	237.63	3077.78	2251.96	4
30	1702.8	1639.79	2068.74	—
31	1877.93	1860.96	1975.3	—
32	867.81	2334.68	2535.1	—
33	1831.49	1713.11	1604.68	—
34	460.69	3274.77	2172.99	—
35	2374.98	3346.98	975.31	—
36	2271.89	3482.97	946.7	—
37	1783.64	1597.99	2261.31	—
38	198.83	3250.45	2445.08	—
39	1494.63	2072.59	2550.51	—
40	1597.03	1921.52	2126.76	—
41	1598.93	1921.08	1623.33	—
42	1243.13	1814.07	3441.07	—
43	2336.31	2640.26	1599.63	—
44	354	3300.12	2373.61	—
45	2144.47	2501.62	591.51	—
46	426.31	3105.29	2057.8	—
47	1507.13	1556.89	1954.51	—
48	343.07	3271.72	2036.94	—
49	2201.94	3196.22	935.53	—
50	2232.43	3077.87	1298.87	—
51	1580.1	1752.07	2463.04	—
52	1962.4	1594.97	1835.95	—
53	1495.18	1957.44	3498.02	—
54	1125.17	1594.39	2937.73	—
55	24.22	3447.31	2145.01	—
56	1269.07	1910.72	2701.97	—
57	1802.07	1725.81	1966.35	—
58	1817.36	1927.4	2328.79	—
59	1860.45	1782.88	1875.13	—

本书分类器的设计都是基于 MATLAB 2016 实现的。MATLAB 软件是实现各种相关算法的大众化软件,2016 版的 MATLAB 已经集成了 100 多种神经网络的工具箱及其他算法的工具箱,使用简单方便。

习题

(1) 什么是模式识别?

(2) 模式识别的基本方法有哪些?

第2章

贝叶斯分类器设计

分类有基于规则的分类(查询)和非规则分类(有指导学习)。贝叶斯分类是非规则分类,它通过训练集(已分类的例子集)训练来归纳出分类器,并利用分类器对未分类的数据进行分类。其基本思想是依据类的概率、概率密度,按照某种准则使分类结果从统计上讲是最佳的。

2.1 贝叶斯决策及贝叶斯公式

2.1.1 贝叶斯决策简介

贝叶斯决策理论就是在不完全情报下,对部分未知的状态用主观概率进行估计,然后用贝叶斯公式对发生的概率进行修正,最后再利用期望值和修正概率做出最优决策。

贝叶斯决策属于风险型决策,决策者虽不能控制客观因素的变化,但却掌握其变化的可能状况及各状况的分布概率,并利用期望值即未来可能出现的平均状况作为决策准则。

贝叶斯决策理论方法是统计模型决策中的一个基本方法,其基本思想如下。

(1) 已知类条件概率密度参数表达式和先验概率。

(2) 利用贝叶斯公式转换成后验概率。

(3) 根据后验概率大小进行决策分类。

2.1.2 贝叶斯公式

若已知总共有 M 类样品,以及各类在 n 维特征空间的统计分布,根据概率知识可以通过样品库得知已知各类别 $\omega_i(i=1,2,\cdots,M)$ 的先验概率 $P(\omega_i)$ 及类条件概率密度函数 $P(\boldsymbol{X}|\omega_i)$。对于待测样品,贝叶斯公式可以计算出该样品分属各类别的概率,叫作后验概率;比较各个后验概率,取 $P(\omega_i|\boldsymbol{X})$ 的最大值,就把 $\boldsymbol{X}$ 归于后验概率最大的那个类。贝叶斯公式为

$$P(\omega_i \mid \boldsymbol{X}) = \frac{P(\boldsymbol{X} \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{M} P(\boldsymbol{X} \mid \omega_j)P(\omega_j)} \tag{2-1}$$

1. 先验概率

先验概率 $P(\omega_i)$ 代表还没有训练数据前 ω_i 拥有的初始概率。$P(\omega_i)$ 反映了我们所拥有的关于 ω_i 是正确分类的背景知识,它是独立于样本的。如果没有这一先验知识,那么可以简单地将每一候选类别赋予相同的先验概率,通常用样本中属于 ω_i 的样本数与总样本数的比值来近似。

2. 后验概率

后验概率是关于随机事件或者不确定性断言的条件概率,是在相关证据或者背景给定并纳入考虑之后的条件概率。后验概率分布就是以未知量作为随机变量的概率分布,并且是在基于实验或者调查所获得的信息上的条件分布。“后验”在这里的意思是考虑相关事件已经被检视并且能够得到一些信息。这种可能性可用 $P(y_m|x)$ 表示。可以利用贝叶斯公式来计算这种条件概率,称为状态的后验概率 $P(y_m|x)$。

$$P(y_m \mid x) = \frac{P(x, y_m)}{P(x)} = \frac{P(x \mid y_m)P(y_m)}{\sum_{m=1}^{M} P(y_m)P(x \mid y_m)} \tag{2-2}$$

$P(y_m|x)$ 表示在 x 出现的条件下样品为 y_m 类的概率。在这里要弄清楚条件概率这个概念。

3. 先验概率与后验概率的区别

先验概率是指根据以往经验和分析得到的概率,它往往作为“由因求果”问题中的“因”出现。后验概率是指在得到“结果”的信息后重新修正的概率,是“执果寻因”问题中的“因”。后验概率是基于新的信息,修正原来的先验概率后所获得的更接近实际情况的概率估计。先验概率和后验概率是相对的。如果以后还有新的信息引入,更新了现在所谓的后验概率,得到新的概率值,那么这个新的概率值被称为后验概率。

2.2 基于最小错误率的贝叶斯决策

基于最小错误率的贝叶斯决策,就是利用贝叶斯公式,按照尽量减少分类错误的原则而得出的一种分类规则,可以使得错误率最小。

2.2.1 基于最小错误率的贝叶斯决策理论

在一般的模式识别问题中,人们的目标往往是尽量减少分类的错误,追求最小的错误率,即求解一种决策规则,使得

$$\min P(e) = \int P(e \mid x)P(x)\mathrm{d}x \tag{2-3}$$

这就是基于最小错误率的贝叶斯决策。

在式(2-3)中,$P(e|x)\geqslant 0$,$P(x)\geqslant 0$ 对于所有的 x 均成立,故 $\min P(e)$ 等同于对所有的

x 最小化 $P(e|x)$，即使后验概率 $P(\omega_i|x)$ 最大化。根据贝叶斯公式

$$P(\omega_i \mid x) = \frac{P(x \mid \omega_i)P(\omega_i)}{P(x)} = \frac{P(x \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{k} P(x \mid \omega_j)P(\omega_j)P(x)} \tag{2-4}$$

在式(2-4)中，对于所有类别，分母都是相同的，所以决策的时候实际上只需要比较分子，即

$$P(x \mid \omega_i)P(\omega_i) = \max_{i=1}^{k} P(x \mid \omega_i)P(\omega_i), \quad x \in \omega_i \tag{2-5}$$

先验概率 $P(\omega_i)$ 和类条件概率密度 $P(x|\omega_i)$ 是已知的。概率密度 $P(x|\omega_i)$ 反映了在 ω_i 类中观察到特征值 x 的相对可能性。

对于多类别决策，错误率的计算量较大，可以转化为计算平均正确率 $P(c)$ 来计算错误率：

$$P(e) = 1 - P(c) = 1 - \sum_{j=1}^{k} P(x \in \mathcal{R} \mid \omega_j)P(\omega_j) = 1 - \sum_{j=1}^{k} P(\omega_j)\int_{\mathcal{R}_j} P(x \mid \omega_j)\mathrm{d}x \tag{2-6}$$

2.2.2 最小错误率贝叶斯分类的计算过程

1. 先期进行的计算

(1) 求出每一类样本的均值。

$$\overline{\boldsymbol{X}_i} = \frac{1}{N_i}\sum_{j=1}^{N_i} X_{ij}, \quad i = 1,2,3,4; \quad j = 0,1,2,\cdots,N_i \tag{2-7}$$

共有 29 个样本，$N_i=29$；分四类，$\omega=4$。

第一类：N_1(样本数目)$=4$；第二类：$N_2=7$；第三类：$N_3=8$；第四类：$N_4=10$。

第一类样本为：

```
A = [864.45   877.88   1418.79   1449.58;1647.31   2031.66   1775.89   1641.58;2665.9
   3071.18   2772.9   3045.12];
```

求出第一类样本的均值为：

```
X1 =
  1.0e + 03
    1.1527
    1.7741
    2.8888
```

(2) 求出每一类样本的协方差矩阵。

求出每一类样本的协方差矩阵 $\boldsymbol{S}_i$，并求出其逆矩阵 $\boldsymbol{S}_i^{-1}$ 和行列式。其中，l 为样本在每一类的序号；j 和 k 为特征值序号；N_i 为每类学习样本中包含的元素个数。

$$\boldsymbol{S}_i = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{bmatrix}$$

$$u_{jk}=\frac{1}{N_i-1}\sum_{i=1}^{N_i}(x_{ij}-\bar{x}_j)(x_{ik}-\bar{x}_k),\quad j,k=1,2,\cdots,n;\quad i=1,2,\cdots,N_i \tag{2-8}$$

第一类样本的协方差矩阵为：

```
S1 =
  1.0e+05 *
    1.0585   -0.2437   0.0990
   -0.2437    0.3333   0.1810
    0.0990    0.1810   0.4027
```

(3) 求第一类样本的协方差矩阵的逆矩阵。

求得第一类样本的协方差矩阵的逆矩阵为：

```
S1_ =
  1.0e-04 *
    0.1419    0.1624   -0.1079
    0.1624    0.5828   -0.3019
   -0.1079   -0.3019    0.4106
```

(4) 求第一类样本的协方差矩阵的行列式值。

求得第一类样本的协方差矩阵的行列式值为：

```
S11 =
    7.1458e+13
```

2. 其他各类求值

第二类样本为：

```
B=[2352.12  2297.28  2092.62  2205.36  2949.16  2802.88  2063.54
   2557.04  3340.14  3177.21  3243.74  3244.44  3017.11  3199.76
   1411.53  535.62   584.32   1202.69  662.42   1984.98  1257.21];
```

其均值为：

```
X2 =
  1.0e+03 *
    2.3947
    3.1113
    1.0913
```

其协方差矩阵为：

```
S2 =
  1.0e+05 *
    1.2035   -0.0627    0.4077
   -0.0627    0.6931   -0.7499
    0.4077   -0.7499    2.8182
```

其协方差矩阵的逆矩阵为：

```
S2_ =
  1.0e-04 *
    0.0877   -0.0081   -0.0149
   -0.0081    0.2033    0.0553
   -0.0149    0.0553    0.0523
```

其协方差矩阵的行列式值为：

```
S22 =
    1.5862e+15
```

第三类样本为：

```
C=[1739.94  1756.77  1803.58  1571.17  1845.59  1692.62  1680.67  1651.52
    1675.15  1652     1583.12  1731.04  1918.81  1867.5   1575.78  1713.28
    2395.96  1514.98  2163.05  1735.33  2226.49  2108.97  1725.1   1570.38];
```

其均值为：

```
X3 =
  1.0e+03 *
    1.7177
    1.7146
    1.9300
```

其协方差矩阵为：

```
S3 =
  1.0e+05 *
    0.0766    0.0150    0.1536
    0.0150    0.1534    0.1294
    0.1536    0.1294    1.1040
```

其协方差矩阵的逆矩阵为：

```
S3_ =
  1.0e-03 *
    0.1813    0.0040   -0.0257
    0.0040    0.0724   -0.0090
   -0.0257   -0.0090    0.0137
```

其协方差矩阵行列式的值为：

```
S33 =
    8.4164e+12
```

第四类样本为：

```
D = [373.3    222.85   401.3    363.34   104.8    499.85   172.78   341.59   291.02
    237.63   3087.05  3059.54  3259.94  3477.95  3389.83  3305.75  3084.49  3076.62
    3095.68  3077.78  2429.47  2002.33  2150.98  2462.86  2421.83  3196.22  2328.65
    2438.63  2088.95  2251.96];
```

其均值为：

```
X4 =
  1.0e + 03 *
    0.3008
    3.1915
    2.3772
```

其协方差矩阵为：

```
S4 =
  1.0e + 05 *
    0.1395    0.0317    0.2104
    0.0317    0.2380    0.2172
    0.2104    0.2172    1.0883
```

其协方差矩阵的逆矩阵为：

```
S4_ =
  1.0e - 03 *
      0.1018    0.0054   - 0.0208
      0.0054    0.0517   - 0.0114
    - 0.0208  - 0.0114     0.0155
```

其协方差矩阵行列式的值为：

```
S44 =
    2.0812e + 13
```

然后计算每类数据的先验概率如下：

```
N = 29;w = 4;n = 3;N1 = 4;N2 = 7;N3 = 8;N4 = 10;
Pw1 = N1/N
Pw1 =
    0.1379
Pw2 = N2/N
Pw2 =
    0.2414
Pw3 = N3/N
Pw3 =
    0.2759
Pw4 = N4/N
Pw4 =
    0.3448
```

至此,前期的计算基本完成。

2.2.3 最小错误率贝叶斯分类的 MATLAB 实现

1. 初始化

初始化程序如下:

```
%输入训练样本数,类别数,特征数,以及属于各类别的样本个数
N = 29;w = 4;n = 3;N1 = 4;N2 = 7;N3 = 8;N4 = 10;
```

2. 参数计算

参数计算程序如下:

```
%计算每一类训练样本的均值
X1 = mean(A')';X2 = mean(B')';X3 = mean(C')';X4 = mean(D')';
%求每一类样本的协方差矩阵
  S1 = cov(A');S2 = cov(B');S3 = cov(C');S4 = cov(D');
%计算协方差矩阵的逆矩阵
  S1_ = inv(S1);S2_ = inv(S2);S3_ = inv(S3);S4_ = inv(S4);
%计算协方差矩阵的行列式
  S11 = det(S1);S22 = det(S2);S33 = det(S3);S44 = det(S4);
%计算训练样本的先验概率
  Pw1 = N1/N;Pw2 = N2/N;Pw3 = N3/N;Pw4 = N4/N; %Priori probability
%计算后验概率:在这里定义了一个循环
for k = 1:30
P1 = -1/2*(sample(k,:)'-X1)'*S1_*(sample(k,:)'-X1)+log(Pw1)-1/2*log(S11);
P2 = -1/2*(sample(k,:)'-X2)'*S2_*(sample(k,:)'-X2)+log(Pw2)-1/2*log(S22);
P3 = -1/2*(sample(k,:)'-X3)'*S3_*(sample(k,:)'-X3)+log(Pw3)-1/2*log(S33);
P4 = -1/2*(sample(k,:)'-X4)'*S4_*(sample(k,:)'-X4)+log(Pw4)-1/2*log(S44);
```

3. MATLAB 完整程序及仿真结果

MATLAB 程序如下:

```
clear;
clc;
N = 29;w = 4;n = 3;N1 = 4;N2 = 7;N3 = 8;N4 = 10;
A = [864.45  877.88  1418.79  1449.58;1647.31  2031.66  1775.89  1641.58;2665.9
    3071.18  2772.9  3045.12];  %A belongs to w1
B = [2352.12  2297.28  2092.62  2205.36  2949.16  2802.88  2063.54
    2557.04  3340.14  3177.21  3243.74  3244.44  3017.11  3199.76
    1411.53  535.62  584.32  1202.69  662.42  1984.98  1257.21];  %B belongs to w2
C = [1739.94  1756.77  1803.58  1571.17  1845.59  1692.62  1680.67  1651.52  1675.15
    1652  1583.12  1731.04  1918.81  1867.5  1575.78  1713.28  2395.96
    1514.98  2163.05  1735.33  2226.49  2108.97  1725.1  1570.38]; %C belongs to w3
```

```
D=[373.3  222.85  401.3  363.34  104.8  499.85  172.78  341.59  291.02  237.63
    3087.05  3059.54  3259.94  3477.95  3389.83  3305.75  3084.49  3076.62  3095.68
    3077.78  2429.47  2002.33  2150.98  2462.86  2421.83  3196.22  2328.65  2438.63
    2088.95  2251.96];                                              % D belongs to w4
 %以上为学习样本数据的输入
  X1=mean(A')';X2=mean(B')';X3=mean(C')';X4=mean(D')';          %求样本均值
  S1=cov(A');S2=cov(B');S3=cov(C');S4=cov(D');                  %求样本协方差矩阵
  S1_=inv(S1);S2_=inv(S2);S3_=inv(S3);S4_=inv(S4);              %求协方差矩阵的逆矩阵
  S11=det(S1);S22=det(S2);S33=det(S3);S44=det(S4);              %求协方差矩阵的行列式
  Pw1=N1/N;Pw2=N2/N;Pw3=N3/N;Pw4=N4/N;                          %先验概率
 %这部分为初始样本数据计算
  sample=[1702.8  1639.79  2068.74
          1877.93 1860.96  1975.3
          867.81  2334.68  2535.1
          1831.49 1713.11  1604.68
          460.69  3274.77  2172.99
          2374.98 3346.98  975.31
          2271.89 3482.97  946.7
          1783.64 1597.99  2261.31
          198.83  3250.45  2445.08
          1494.63 2072.59  2550.51
          1597.03 1921.52  2126.76
          1598.93 1921.08  1623.33
          1243.13 1814.07  3441.07
          2336.31 2640.26  1599.63
          354     3300.12  2373.61
          2144.47 2501.62  591.51
          426.31  3105.29  2057.8
          1507.13 1556.89  1954.51
          343.07  3271.72  2036.94
          2201.94 3196.22  935.53
          2232.43 3077.87  1298.87
          1580.1  1752.07  2463.04
          1962.4  1594.97  1835.95
          1495.18 1957.44  3498.02
          1125.17 1594.39  2937.73
          24.22   3447.31  2145.01
          1269.07 1910.72  2701.97
          1802.07 1725.81  1966.35
          1817.36 1927.4   2328.79
          1860.45 1782.88  1875.13];
 %这部分为测试数据输入
for k=1:30
P1=-1/2*(sample(k,:)'-X1)'*S1_*(sample(k,:)'-X1)+log(Pw1)-1/2*log(S11);
 %第一类的判别函数
P2=-1/2*(sample(k,:)'-X2)'*S2_*(sample(k,:)'-X2)+log(Pw2)-1/2*log(S22);
 %第二类的判别函数
P3=-1/2*(sample(k,:)'-X3)'*S3_*(sample(k,:)'-X3)+log(Pw3)-1/2*log(S33);
 %第三类的判别函数
P4=-1/2*(sample(k,:)'-X4)'*S4_*(sample(k,:)'-X4)+log(Pw4)-1/2*log(S44);
```

```
%第四类的判别函数
P = [P1 P2 P3 P4]
Pmax = max(P)
  if P1 == max(P)
    w = 1
      plot3(sample(k,1),sample(k,2),sample(k,3),'ro');grid on;hold on;
  elseif P2 == max(P)
    w = 2
      plot3(sample(k,1),sample(k,2),sample(k,3),'b>');grid on;hold on;
  elseif P3 == max(P)
      w = 3
      plot3(sample(k,1),sample(k,2),sample(k,3),'g+');grid on;hold on;
  elseif P4 == max(P)
      w = 4
      plot3(sample(k,1),sample(k,2),sample(k,3),'y*');grid on;hold on;
  else
    return        %判别函数最大值对应的类别
    end
end
```

运行程序得出如图 2-1 所示的测试数据分类图界面。

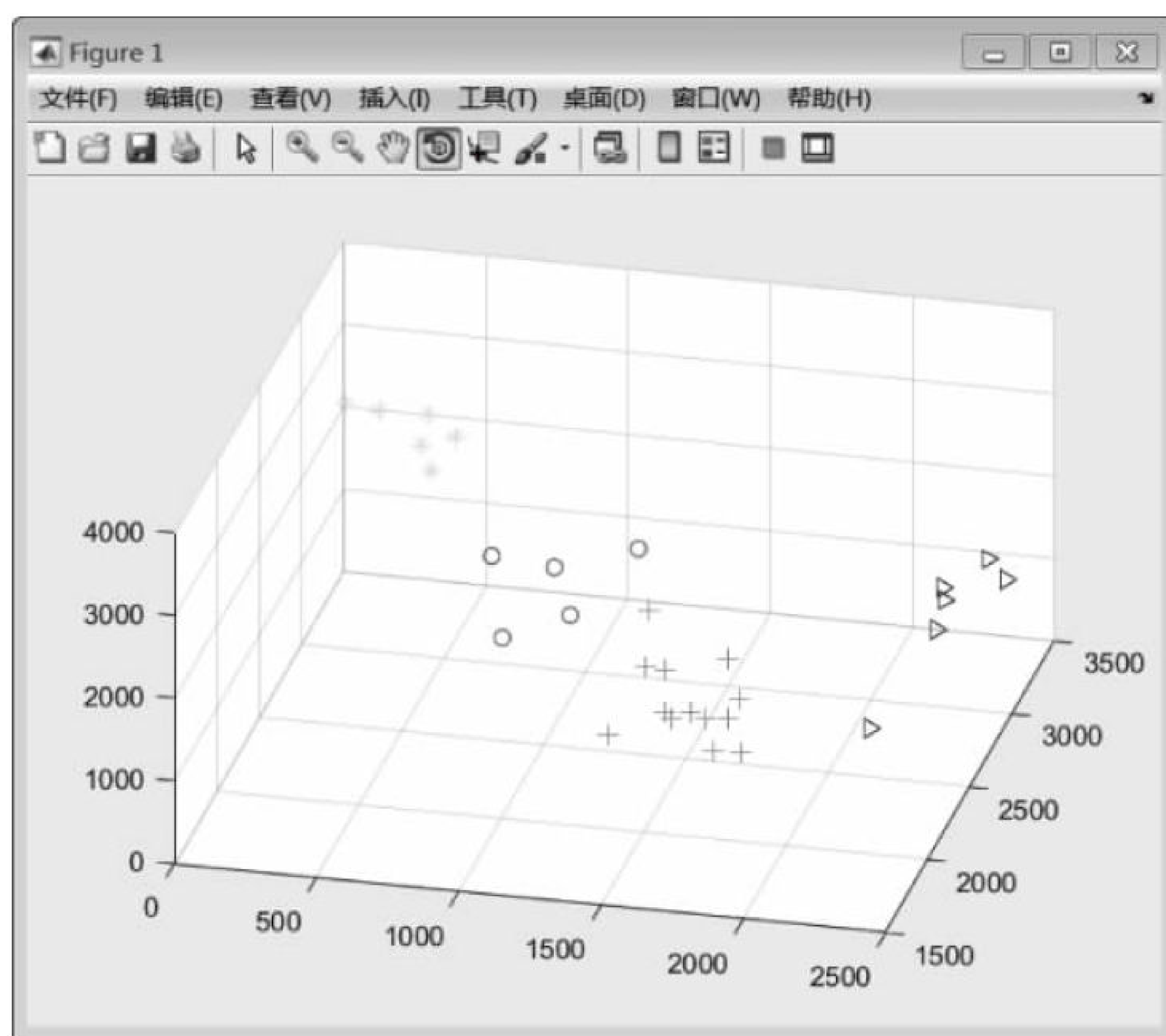


图 2-1 测试数据分类图界面

从图 2-5 中可以看出数据分类效果比较好。

MATLAB 程序运行结果如下：

```
P =
-34.7512   -37.7619   -16.6744   -171.1604
```

```
Pmax =
 -16.6744
w =
3
P =
 -49.5808   -32.0756   -19.1319   -185.7206
Pmax =
 -19.1319
w =
3
P =
 -32.5380   -36.8429   -105.8245   -48.9684
Pmax =
 -32.5380
w =
1
P =
 -61.5273   -36.6998   -19.0123   -196.0725
Pmax =
 -19.0123
w =
3
P =
 -107.6977   -42.9982   -244.6344   -19.1425
Pmax =
 -19.1425
w =
4
P =
 -323.1237   -19.3722   -192.5329   -315.7399
Pmax =
 -19.3722
w =
2
P =
 -344.0690   -20.1602   -197.4786   -298.5021
Pmax =
 -20.1602
w =
2
P =
 -28.8735   -37.9474   -17.5637   -182.7101
Pmax =
 -17.5637
w =
3
P =
 -84.2886   -50.7629   -316.2804   -17.1190
Pmax =
 -17.1190
```

```
w =
4
P =
 - 29.6605   - 31.8272   - 29.1895   - 112.1975
Pmax =
 - 29.1895
w =
3
P =
 - 39.9950   - 32.5544   - 19.4480   - 138.2933
Pmax =
 - 19.4480
w =
3
P =
 - 65.6213   - 33.2003   - 19.1755   - 148.7788
Pmax =
 - 19.1755
w =
3
P =
 - 23.1508   - 42.2485   - 69.4564   - 108.1711
Pmax =
 - 23.1508
w =
1
P =
 - 150.6823   - 20.5669   - 92.9234   - 261.7163
Pmax =
 - 20.5669
w =
2
P =
 - 95.2729   - 47.3863   - 277.7827   - 16.8863
Pmax =
 - 16.8863
w =
4
P =
 - 235.4394   - 25.0041   - 92.9044   - 273.8238
Pmax =
 - 25.0041
w =
2
P =
 - 98.6750   - 41.1397   - 233.0440   - 18.6408
Pmax =
 - 18.6408
w =
```

```
4
P =
 - 34.3114   - 41.4903   - 21.3935   - 152.9500
Pmax =
 - 21.3935
w =
3
P =
 - 114.2256   - 43.9679   - 269.1704   - 18.1769
Pmax =
 - 18.1769
w =
4
P =
 - 293.2196   - 19.1168   - 152.2279   - 273.4275
Pmax =
 - 19.1168
w =
2
P =
 - 231.6066   - 19.1683   - 129.1233   - 256.2709
Pmax =
 - 19.1683
w =
2
P =
 - 24.4899   - 35.9915   - 21.5651   - 142.4473
Pmax =
 - 21.5651
w =
3
P =
 - 47.4226   - 38.2727   - 22.5483   - 219.5498
Pmax =
 - 22.5483
w =
3
P =
 - 22.7587   - 38.1842   - 44.9330   - 118.0112
Pmax =
 - 22.7587
w =
1
P =
 - 19.2874   - 44.7372   - 72.1963   - 112.7623
Pmax =
 - 19.2874
w =
1
```

```
P =
-117.7652   -53.9295   -379.5944   -21.3597
Pmax =
-21.3597
w =
4
P =
-20.5508   -36.8251   -47.0716   -98.7984
Pmax =
-20.5508
w =
1
P =
-43.0681   -35.3828   -16.7483   -181.9830
Pmax =
-16.7483
w =
3
P =
-36.4518   -31.0477   -18.0932   -165.2229
Pmax =
-18.0932
w =
3
P =
-50.6918   -34.0120   -18.4786   -  189.7615
Pmax =
-18.4786
w =
3
```

表2-1所示为待测样本分类表。

表 2-1 待测样本分类表

A	B	C	P1	P2	P3	P4	类 别
1702.8	1639.79	2068.74	−34.7512	−37.7619	−16.6744	−171.1604	3
1877.93	1860.96	1975.3	−49.5808	−32.0756	−19.1319	−185.7206	3
867.81	2334.68	2535.1	−32.5380	−36.8429	−105.8245	−48.9684	1
1831.49	1713.11	1604.68	−61.5273	−36.6998	−19.0123	−196.0725	3
460.69	3274.77	2172.99	−107.6977	−42.9982	−244.6344	−19.1425	4
2374.98	3346.98	975.31	−323.1237	−19.3722	−192.5329	−315.7399	2
2271.89	3482.97	946.7	−344.0690	−20.1602	−197.4786	−298.5021	2
1783.64	1597.99	2261.31	−28.8735	−37.9474	−17.5637	−182.7101	3
198.83	3250.45	2445.08	−84.2886	−50.7629	−316.2804	−17.1190	4
1494.63	2072.59	2550.51	−29.6605	−31.8272	−29.1895	−112.1975	3
1597.03	1921.52	2126.76	−39.9950	−32.5544	−19.4480	−138.2933	3
1598.93	1921.08	1623.33	−65.6213	−33.2003	−19.1755	−148.7788	3
1243.13	1814.07	3441.07	−23.1508	−42.2485	−69.4564	−108.1711	1

续表

A	B	C	P1	P2	P3	P4	类 别
2336.31	2640.26	1599.63	−150.6823	−20.5669	−92.9234	−261.7163	2
354	3300.12	2373.61	−95.2729	−47.3863	−277.7827	−16.8863	4
2144.47	2501.62	591.51	−235.4394	−25.0041	−92.9044	−273.8238	2
426.31	3105.29	2057.8	−98.6750	−41.1397	−233.0440	−18.6408	4
1507.13	1556.89	1954.51	−34.3114	−41.4903	−21.3935	−152.9500	3
343.07	3271.72	2036.94	−114.2256	−43.9679	−269.1704	−18.1769	4
2201.94	3196.22	935.53	−293.2196	−19.1168	−152.2279	−273.4275	2
2232.43	3077.87	1298.87	−231.6066	−19.1683	−129.1233	−256.2709	2
1580.1	1752.07	2463.04	−24.4899	−35.9915	−21.5651	−142.4473	3
1962.4	1594.97	1835.95	−47.4226	−38.2727	−22.5483	−219.5498	3
1495.18	1957.44	3498.02	−22.7587	−38.1842	−44.9330	−118.0112	1
1125.17	1594.39	2937.73	−19.2874	−44.7372	−72.1963	−112.7623	1
24.22	3447.31	2145.01	−117.7652	−53.9295	−379.5944	−21.3597	4
1269.07	1910.72	2701.97	−20.5508	−36.8251	−47.0716	−98.7984	1
1802.07	1725.81	1966.35	−43.0681	−35.3828	−16.7483	−181.9830	3
1817.36	1927.4	2328.79	−36.4518	−31.0477	−18.0932	−165.2229	3
1860.45	1782.88	1875.13	−50.6918	−34.0120	−18.4786	−189.7615	3

对比正确分类后发现，只有一组数据(1494.63,2072.59,2550.51)与正确分类有出入，该分类是第三类(ω_3)，但正确分类为第一类(ω_1)。

反过来验证一下学习样本，程序不变，只在数据输入时改成学习样本，循环次数做一下调整，得到的结果如图2-2所示。

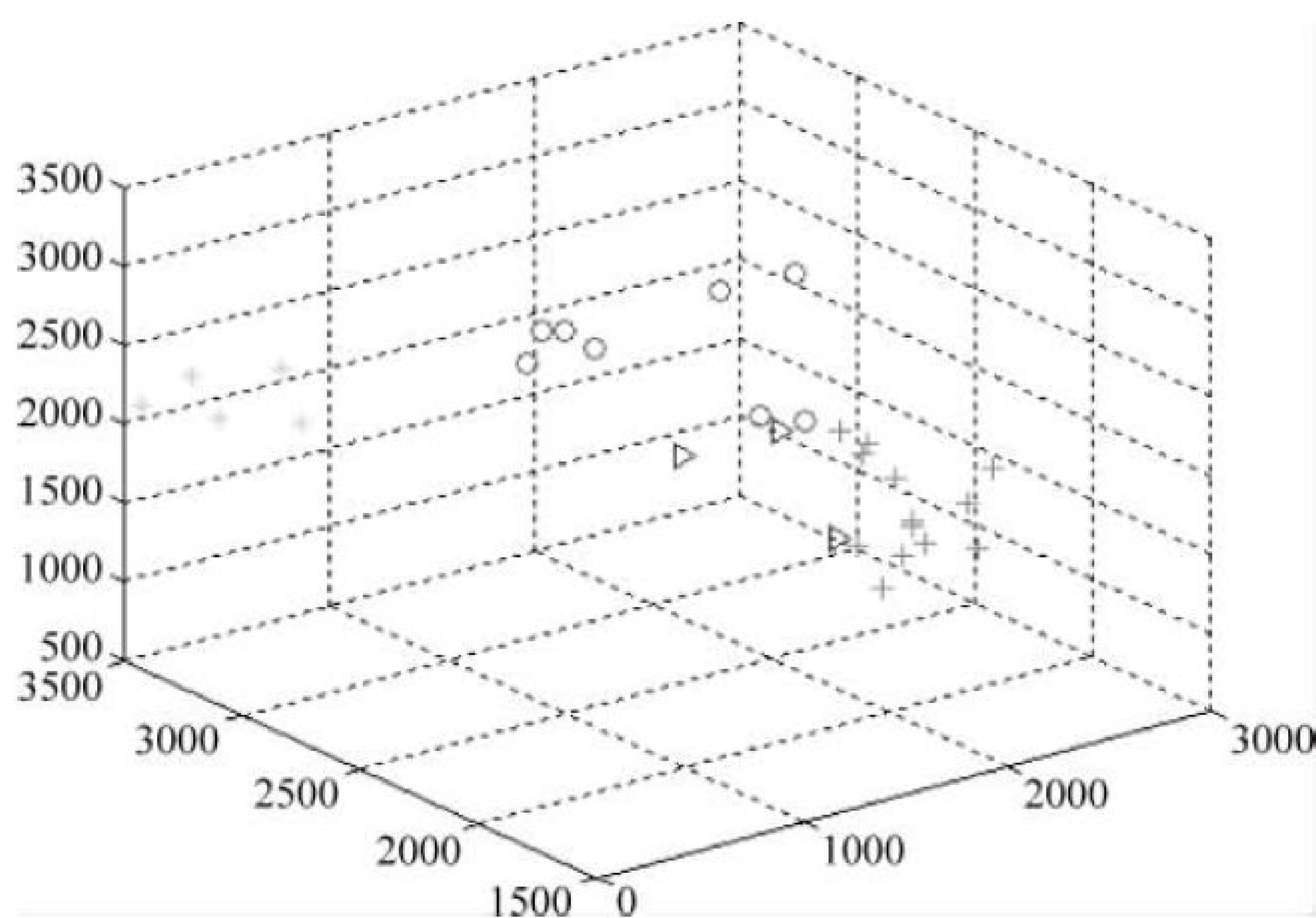


图2-2 训练样本分类图

可以看到验证出的数据与原始学习样本的分类是吻合的，因此可以判定本例的最小错误率贝叶斯判别方法基本正确。

2.2.4 结论

从理论上讲，依据贝叶斯理论所设计的分类器应该具有最优的性能，如果所有的模式识

别问题都可以这样来解决，那么模式识别问题就成了一个简单的计算问题，但是实际问题往往更复杂。贝叶斯决策理论要求两个前提，一个是分类类别数目已知；另一个是类条件概率密度和先验概率已知。前者很容易解决，而后者通常就不容易满足了。基于贝叶斯决策的分类器设计方法是在已知类条件概率密度的情况下讨论的，贝叶斯判别函数中的类条件概率密度是利用样本来估计的，估计出来的类条件概率密度函数可能是线性函数，也可能是各种各样的非线性函数。这种设计判别函数的思路，在用样本估计之前，是不知道判别函数是线性函数还是别的什么函数的，而且有时候受样本空间大小、维数等影响，类条件概率密度函数更加难以确定。

2.3 最小风险贝叶斯决策

2.3.1 最小风险贝叶斯决策理论

决策理论是为了实现特定的目标，根据客观的可能性，在占有一定信息和经验的基础上，借助一定的工具、技巧和方法，对影响未来目标实现的诸因素进行准确的计算和判断优选后，对未来行动做出决定。在某些情况下，引入风险的概念以求风险最小的决策更为合理。例如对癌细胞的识别，识别的正确与否直接关系到病人的身体健康。风险的概念常与损失相联系。当参数的真值和决策结果不一致会带来损失时，这种损失作为参数的真值和决策结果的函数，称为损失函数。损失函数的期望值称为风险函数。为了分析引入损失函数 $\lambda(\alpha_i,\omega_j)(i=1,2,\cdots,a;\ j=1,2,\cdots,m)$，这个函数表示当处于状态 ω_j 时采取决策为 α_i 所带来的损失。在决策论中，常以决策表一目了然地表示各种情况下的决策损失，我们用表 2-2 来描述。这是在已知先验概率 $P(\omega_j)$ 及类条件概率密度 $P(\boldsymbol{X}|\omega_j)(j=1,2,\cdots,m)$ 的条件下进行讨论的。

表 2-2 贝叶斯决策表

决　策	状　态					
	ω_1	ω_2	…	ω_j	…	ω_m
α_1	$\lambda(\alpha_1,\omega_1)$	$\lambda(\alpha_1,\omega_2)$	…	$\lambda(\alpha_1,\omega_j)$	…	$\lambda(\alpha_1,\omega_m)$
α_2	$\lambda(\alpha_2,\omega_1)$	$\lambda(\alpha_2,\omega_2)$	…	$\lambda(\alpha_2,\omega_j)$	…	$\lambda(\alpha_2,\omega_m)$
⋮						
α_i	$\lambda(\alpha_i,\omega_1)$	$\lambda(\alpha_i,\omega_2)$	…	$\lambda(\alpha_i,\omega_j)$	…	$\lambda(\alpha_i,\omega_m)$
⋮						
α_a	$\lambda(\alpha_a,\omega_1)$	$\lambda(\alpha_a,\omega_2)$	…	$\lambda(\alpha_a,\omega_j)$	…	$\lambda(\alpha_a,\omega_m)$

根据贝叶斯公式，后验概率为

$$P(\omega_j \mid \boldsymbol{X}) = \frac{P(\boldsymbol{X} \mid \omega_j)P(\omega_j)}{\sum_{i=1}^{a} P(\boldsymbol{X} \mid \omega_i)P(\omega_i)}, \quad j = 1,2,\cdots,m \tag{2-9}$$

当引入"损失"的概念后考虑错判所造成的损失时，就不能只根据后验概率的大小做决策，而必须考虑所采取的决策是否损失为最小。对于给定的 $\boldsymbol{X}$，如果采取决策 $\alpha_i(i=1,2,\cdots,a)$，λ 可以在 m 个 $\lambda(\alpha_i,\omega_j)$，$j=1,2,\cdots,m$ 当中任取一个，其相应概率密度函数为 $P(\omega_j|\boldsymbol{X})$。因此在采取决策 α_i 情况下的条件期望损失为

$$R(\alpha_i \mid \boldsymbol{X}) = E[\lambda(\alpha_i,\omega_j)] = \sum_{j=1}^{m}\lambda(\alpha_i,\omega_j)P(\omega_j \mid \boldsymbol{X}),\quad i=1,2,\cdots,a \tag{2-10}$$

在决策论中又把采取决策 α_i 的条件期望损失 $R(\alpha_i|\boldsymbol{X})$ 称为条件风险。由于 $\boldsymbol{X}$ 是随机向量的观察值，对于 $\boldsymbol{X}$ 的不同观察值，采取 α_i 决策时，其条件风险的大小是不同的。所以究竟采取哪一种决策将随 $\boldsymbol{X}$ 的取值而定。决策 α 可以被看成随机向量 $\boldsymbol{X}$ 的函数，记为 $\alpha(\boldsymbol{X})$，这里定义期望风险为

$$R = \int R(\alpha(\boldsymbol{X}) \mid \boldsymbol{X})P(\boldsymbol{X})\mathrm{d}x \tag{2-11}$$

式中，$\mathrm{d}x$ 是特征空间的体积元，积分在整个特征空间进行。期望风险 R 反映对整个特征空间所有 $\boldsymbol{X}$ 的取值都采取相应的决策 $\alpha(\boldsymbol{X})$所带来的平均风险；而条件风险 $R(\alpha_i|\boldsymbol{X})$只是反映了对某一 $\boldsymbol{X}$ 的取值采取决策 α_i 所带来的风险。显然，需要采取一系列决策 $\alpha(\boldsymbol{X})$使期望风险 R 最小。在考虑错判带来的损失时，我们希望损失最小。如果在采取每一个决策或行动时，都使其风险最小，则对所有的 $\boldsymbol{X}$ 做出决策时，其期望风险也必然最小，这样的决策就是最小风险贝叶斯决策。

最小风险贝叶斯决策规则为：如果

$$R(\alpha_k \mid \boldsymbol{X}) = \min R(\alpha_i \mid \boldsymbol{X}),\quad i=1,2,\cdots,a \tag{2-12}$$

则有 $\alpha=\alpha(k)$（即采取决策 α_k）。对于实际问题，可按下列步骤进行最小风险贝叶斯决策。

(1) 在已知 $P(\omega_j)$，$P(\boldsymbol{X}|\omega_j)$，$j=1,2,\cdots,m$，并给出待识别的 $\boldsymbol{X}$ 的情况下，根据贝叶斯公式可以计算出后验概率，见式(2-9)。

(2) 利用计算出的后验概率及决策表，按式(2-10)，计算出 $\alpha_i(i=1,2,\cdots,a)$的条件风险 $R(\alpha_i|\boldsymbol{X})$。

(3) 对步骤(2)中得到的 a 个条件风险值 $R(\alpha_i|\boldsymbol{X})(i=1,2,\cdots,a)$进行比较，找出使条件风险最小的决策 α_k，根据式(2-12)，则 α_k 就是最小风险贝叶斯决策。

应指出，最小风险贝叶斯决策除了要符合实际情况的先验概率 $P(\omega_j)$及类条件概率密度 $P(\boldsymbol{X}|\omega_j)(j=1,2,\cdots,m)$外，还必须要有适合的损失函数 $\lambda(\alpha_i,\omega_j)(i=1,2,\cdots,a;\ j=1,2,\cdots,m)$。实际工作中要列出合适的决策表很不容易，往往要根据研究的具体问题，通过分析错误决策所造成的损失的严重程度，与有关专家共同商讨来确定。

2.3.2 最小错误率与最小风险的贝叶斯决策比较

错误率最小的贝叶斯决策规则与风险最小的贝叶斯决策规则有着某种联系。这里再讨论一下两者的关系。首先设损失函数为

$$\lambda(\alpha_i,\omega_j) = \begin{cases} 0, & i=j \\ 1, & i \neq j \end{cases} \quad i,j=1,2,\cdots,m \tag{2-13}$$

式中，假设对于 m 类只有 m 个决策，即不考虑"拒绝"的情况，对于正确决策即 $i=j$，$\lambda(\alpha_i,\omega_j)=$

0,就是说没有损失;对于任何错误决策,其损失为1,这样定义的损失函数称为0-1损失函数。此时条件风险为

$$R(\alpha_i \mid \boldsymbol{X}) = \sum_{j=1}^{m} \lambda(\alpha_i, \omega_j) P(\omega_j \mid \boldsymbol{X}) = \sum_{j=1, j \neq i}^{m} P(\omega_j \mid \boldsymbol{X}), \quad i = 1, 2, \cdots, m \tag{2-14}$$

式中,$\sum_{j=1, j \neq i}^{m} P(\omega_j \mid \boldsymbol{X})$表示对$\boldsymbol{X}$采取决策$\omega_i$(此时$\omega_j$就相当于$\alpha_i$)的条件错误概率。所以在采取0-1损失函数时,使

$$R(\alpha_k \mid \boldsymbol{X}) = \min P(\alpha_i \mid \boldsymbol{X}), \quad i = 1, 2, \cdots, m \tag{2-15}$$

得最小风险贝叶斯决策,就等价于式(2-16)的最小错误率贝叶斯决策。

$$\sum_{j=1, j \neq i}^{m} P(\omega_j \mid \boldsymbol{X}) = \min \sum_{j=1, j \neq i}^{m} P(\omega_j \mid \boldsymbol{X}), \quad i = 1, 2, \cdots, m \tag{2-16}$$

由此可见,最小错误率贝叶斯决策就是在采用0-1损失函数条件下的最小风险贝叶斯决策,即前者是后者的特例。

2.3.3 贝叶斯算法的计算过程

(1) 输入类数M,特征数n,待分样本数m。

(2) 输入训练样本数N和训练集矩阵$\boldsymbol{X}(N \times n)$,并计算有关参数。

(3) 计算待分析样本的后验概率。

(4) 若按最小风险原则分类,则输入各值,计算各样本属于各类时的风险并判定各样本类别。

2.3.4 最小风险贝叶斯分类的MATLAB实现

1. 初始化

初始化程序如下:

```
%输入训练样本数,类别数,特征数,以及属于各类别的样本个数
N = 29;w = 4;n = 3;N1 = 4;N2 = 7;N3 = 8;N4 = 10;
```

2. 参数计算

参数计算程序如下:

```
%计算每一类训练样本的均值
X1 = mean(A')';X2 = mean(B')';X3 = mean(C')';X4 = mean(D')';
%求每一类样本的协方差矩阵
  S1 = cov(A');S2 = cov(B');S3 = cov(C');S4 = cov(D');
%计算协方差矩阵的逆矩阵
  S1_ = inv(S1);S2_ = inv(S2);S3_ = inv(S3);S4_ = inv(S4);
```

```
%计算协方差矩阵的行列式
  S11 = det(S1);S22 = det(S2);S33 = det(S3);S44 = det(S4);
%计算训练样本的先验概率
  Pw1 = N1/N;Pw2 = N2/N;Pw3 = N3/N;Pw4 = N4/N; % Priori probability
%定义损失函数
loss = ones(4) - diag(diag(ones(4))); % define the riskloss function (4 * 4)
%计算后验概率:在这里定义了一个循环
for k = 1:30
P1 = -1/2 * (sample(k,:)' - X1)' * S1_ * (sample(k,:)' - X1) + log(Pw1) - 1/2 * log(S11);
P2 = -1/2 * (sample(k,:)' - X2)' * S2_ * (sample(k,:)' - X2) + log(Pw2) - 1/2 * log(S22);
P3 = -1/2 * (sample(k,:)' - X3)' * S3_ * (sample(k,:)' - X3) + log(Pw3) - 1/2 * log(S33);
P4 = -1/2 * (sample(k,:)' - X4)' * S4_ * (sample(k,:)' - X4) + log(Pw4) - 1/2 * log(S44);
%计算采取决策 αi 所带来的风险
risk1 = loss(1,1) * P1 + loss(1,2) * P2 + loss(1,3) * P3 + loss(1,4) * P4;
risk2 = loss(2,1) * P1 + loss(2,2) * P2 + loss(2,3) * P3 + loss(2,4) * P4;
risk3 = loss(3,1) * P1 + loss(3,2) * P2 + loss(3,3) * P3 + loss(3,4) * P4;
risk4 = loss(4,1) * P1 + loss(4,2) * P2 + loss(4,3) * P3 + loss(4,4) * P4;
risk = [risk1 risk2 risk3 risk4]
%找出最小风险值
minriskloss = min(risk) % find the least riskloss
```

3. 完整程序及仿真结果

程序代码如下:

```
%minimum bayes risk classifier(文件说明)
%清空工作空间及命令行
clear;
clc;
%输入训练样本数,类别数,特征数,以及属于各类别的样本个数
N = 29;w = 4;n = 3;N1 = 4;N2 = 7;N3 = 8;N4 = 10;
%输入训练样本数据:分别输入1,2,3,4类的矩阵 A,B,C,D
A = [864.45  877.88  1418.79  1449.58
    1647.31  2031.66  1775.89  1641.58
    2665.9  3071.18  2772.9  3045.12]; %A belongs to w1
B = [2352.12  2297.28  2092.62  2205.36  2949.16  2802.88  2063.54
    2557.04  3340.14  3177.21  3243.74  3244.44  3017.11  3199.76
    1411.53  535.62  584.32  1202.69  662.42  1984.98  1257.21]; %B belongs to w2
C = [1739.94  1756.77  1803.58  1571.17  1845.59  1692.62  1680.67  1651.52  1675.15
    1652  1583.12  1731.04  1918.81  1867.5  1575.78  1713.28  2395.96
    1514.98  2163.05  1735.33  2226.49  2108.97  1725.1  1570.38]; %C belongs to w3
D = [373.3  222.85  401.3  363.34  104.8  499.85  172.78  341.59  291.02  237.63  3087.05
    3059.54  3259.94  3477.95  3389.83  3305.75  3084.49  3076.62  3095.68  3077.78
    2429.47  2002.33  2150.98  2462.86  2421.83  3196.22  2328.65  2438.63  2088.95
2251.96]; %D belongs to w4
%计算每一类训练样本的均值
```

```
X1 = mean(A')';X2 = mean(B')';X3 = mean(C')';X4 = mean(D')';   % mean of training samples for
                                                               % each category
%求每一类样本的协方差矩阵
  S1 = cov(A');S2 = cov(B');S3 = cov(C');S4 = cov(D');         % covariance matrix of training
                                                               % samples for each type
%计算协方差矩阵的逆矩阵
  S1_ = inv(S1);S2_ = inv(S2);S3_ = inv(S3);S4_ = inv(S4);     % inverse matrix of training samples
                                                               % for each type
 %计算协方差矩阵的行列式
 S11 = det(S1);S22 = det(S2);S33 = det(S3);S44 = det(S4);      % determinant of covariance matrix
 %计算训练样本的先验概率
 Pw1 = N1/N;Pw2 = N2/N;Pw3 = N3/N;Pw4 = N4/N;                  % Priori probability
  sample = [1702.8    1639.79    2068.74
          1877.93    1860.96    1975.3
          867.81     2334.68    2535.1
          1831.49    1713.11    1604.68
          460.69     3274.77    2172.99
          2374.98    3346.98    975.31
          2271.89    3482.97    946.7
          1783.64    1597.99    2261.31
          198.83     3250.45    2445.08
          1494.63    2072.59    2550.51
          1597.03    1921.52    2126.76
          1598.93    1921.08    1623.33
          1243.13    1814.07    3441.07
          2336.31    2640.26    1599.63
          354        3300.12    2373.61
          2144.47    2501.62    591.51
          426.31     3105.29    2057.8
          1507.13    1556.89    1954.51
          343.07     3271.72    2036.94
          2201.94    3196.22    935.53
          2232.43    3077.87    1298.87
          1580.1     1752.07    2463.04
          1962.4     1594.97    1835.95
          1495.18    1957.44    3498.02
          1125.17    1594.39    2937.73
          24.22      3447.31    2145.01
          1269.07    1910.72    2701.97
          1802.07    1725.81    1966.35
          1817.36    1927.4     2328.79
          1860.45    1782.88    1875.13];
%Posterior probability as the following
%定义损失函数
loss = ones(4) - diag(diag(ones(4)));   % define the riskloss function (4 * 4)
plot(loss); grid on;
xlabel('type');ylabel('Loss function value');
```

```
%计算后验概率：在这里定义了一个循环
for k = 1:30
P1 = -1/2 * (sample(k,:)' - X1)' * S1_ * (sample(k,:)' - X1) + log(Pw1) - 1/2 * log(S11);
P2 = -1/2 * (sample(k,:)' - X2)' * S2_ * (sample(k,:)' - X2) + log(Pw2) - 1/2 * log(S22);
P3 = -1/2 * (sample(k,:)' - X3)' * S3_ * (sample(k,:)' - X3) + log(Pw3) - 1/2 * log(S33);
P4 = -1/2 * (sample(k,:)' - X4)' * S4_ * (sample(k,:)' - X4) + log(Pw4) - 1/2 * log(S44);
%计算采取决策 αi 所带来的风险
risk1 = loss(1,1) * P1 + loss(1,2) * P2 + loss(1,3) * P3 + loss(1,4) * P4;
risk2 = loss(2,1) * P1 + loss(2,2) * P2 + loss(2,3) * P3 + loss(2,4) * P4;
risk3 = loss(3,1) * P1 + loss(3,2) * P2 + loss(3,3) * P3 + loss(3,4) * P4;
risk4 = loss(4,1) * P1 + loss(4,2) * P2 + loss(4,3) * P3 + loss(4,4) * P4;
risk = [risk1 risk2 risk3 risk4]
%找出最小风险值
minriskloss = min(risk)   %find the least riskloss
%返回测试样本的所属类别
%return the category of the least riskloss as following
  if risk1 == min(risk)
    w = 1
  elseif risk2 == min(risk)
    w = 2
  elseif risk3 == min(risk)
      w = 3
  elseif risk4 == min(risk)
    w = 4
  else
    return
     end
end
```

运行程序得到的损失函数的矩阵为：

```
loss =
      0    1    1    1
      1    0    1    1
      1    1    0    1
      1    1    1    0
```

得到如表 2-3 所示的贝叶斯决策表。

表 2-3　贝叶斯决策表

决　策	类　别			
	ω_1	ω_2	ω_3	ω_4
α_1	0	1	1	1
α_2	1	0	1	1
α_3	1	1	0	1
α_4	1	1	1	0

损失函数的函数图界面如图 2-3 所示。

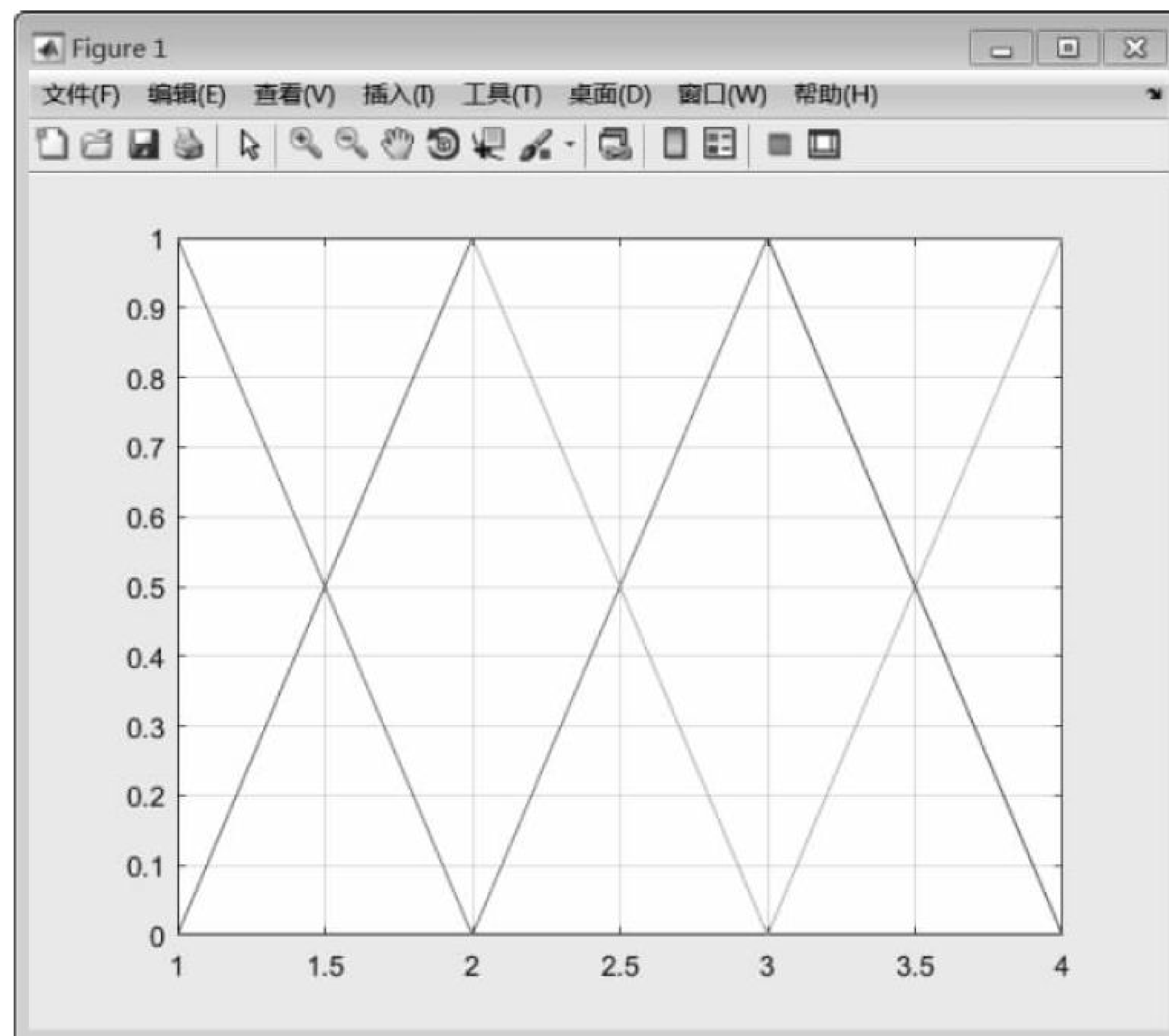


图 2-3 损失函数的函数图界面

继续运行程序,MATLAB 命令窗口显示的结果如下:

```
risk =
-225.5967   -222.5860   -243.6736   -89.1876
minriskloss =
-243.6736
w =
3
risk =
-236.9281   -254.4333   -267.3770   -100.7884
minriskloss =
-267.3770
w =
3
risk =
-191.6357   -187.3309   -118.3493   -175.2054
minriskloss =
-191.6357
w =
1
risk =
-251.7846   -276.6121   -294.2996   -117.2393
minriskloss =
-294.2996
w =
3
```

```
risk =
 -306.7751  -371.4747  -169.8384  -395.3304
minriskloss =
 -395.3304
w =
4
risk =
 -527.6450  -831.3965  -658.2358  -535.0288
minriskloss =
 -831.3965
w =
2
risk =
 -516.1409  -840.0497  -662.7313  -561.7078
minriskloss =
 -840.0497
w =
2
risk =
 -238.2212  -229.1473  -249.5309  -84.3845
minriskloss =
 -249.5309
w =
3
risk =
 -384.1623  -417.6881  -152.1706  -451.3320
minriskloss =
 -451.3320
w =
4
risk =
 -173.2142  -171.0475  -173.6852  -90.6772
minriskloss =
 -173.6852
w =
3
risk =
 -190.2958  -197.7363  -210.8427  -91.9974
minriskloss =
 -210.8427
w =
3
risk =
 -201.1545  -233.5756  -247.6003  -117.9971
minriskloss =
 -247.6003
w =
3
```

```
risk =
-219.8759   -200.7782   -173.5703   -134.8556
minriskloss =
-219.8759
w =
1
risk =
-375.2066   -505.3220   -432.9655   -264.1726
minriskloss =
-505.3220
w =
2
risk =
-342.0554   -389.9420   -159.5456   -420.4420
minriskloss =
-420.4420
w =
4
risk =
-391.7323   -602.1676   -534.2673   -353.3480
minriskloss =
-602.1676
w =
2
risk =
-292.8245   -350.3599   -158.4555   -372.8588
minriskloss =
-372.8588
w =
4
risk =
-215.8337   -208.6549   -228.7517   -97.1952
minriskloss =
-228.7517
w =
3
risk =
-331.3153   -401.5729   -176.3704   -427.3639
minriskloss =
-427.3639
w =
4
risk =
-444.7721   -718.8749   -585.7638   -464.5642
minriskloss =
-718.8749
w =
2
```

```
risk =
 -404.5625  -617.0008  -507.0459  -379.8982
minriskloss =
 -617.0008
w =
2
risk =
 -200.0040  -188.5023  -202.9287  -82.0465
minriskloss =
 -202.9287
w =
3
risk =
 -280.3708  -289.5207  -305.2451  -108.2437
minriskloss =
 -305.2451
w =
3
risk =
 -201.1284  -185.7029  -178.9540  -105.8759
minriskloss =
 -201.1284
w =
1
risk =
 -229.6958  -204.2460  -176.7869  -136.2208
minriskloss =
 -229.6958
w =
1
risk =
 -454.8835  -518.7193  -193.0544  -551.2891
minriskloss =
 -551.2891
w =
4
risk =
 -182.6951  -166.4209  -156.1743  -104.4475
minriskloss =
 -182.6951
w =
1
risk =
 -234.1142  -241.7995  -260.4340  -95.1993
minriskloss =
 -260.4340
w =
3
risk =
 -214.3639  -219.7679  -232.7225  -85.5928
```

```
minriskloss =
 - 232.7225
w =
3
risk =
 - 242.2520   - 258.9318   - 274.4652   - 103.1823
minriskloss =
 - 274.4652
w =
3
```

运行程序得到的结果整理如表 2-4 所示。

表 2-4 待测样本分类表

待测样本特征			属于 1 类风险	属于 2 类风险	属于 3 类风险	属于 4 类风险	最小风险	所属类别
1702.8	1639.79	2068.74	**−225.5967**	**−222.5860**	**−243.6736**	**−89.1876**	**−243.6736**	**3**
1877.93	1860.96	1975.3	**−236.9281**	**−254.4333**	**−267.3770**	**−100.7884**	**−267.3770**	**3**
867.81	2334.68	2535.1	**−191.6357**	**−187.3309**	**−118.3493**	**−175.2054**	**−191.6357**	**1**
1831.49	1713.11	1604.68	**−251.7846**	**−276.6121**	**−294.2996**	**−117.2393**	**−294.2996**	**3**
460.69	3274.77	2172.99	**−306.7751**	**−371.4747**	**−169.8384**	**−395.3304**	**−395.3304**	**4**
2374.98	3346.98	975.31	**−527.6450**	**−831.3965**	**−658.2358**	**−535.0288**	**−831.3965**	**2**
2271.89	3482.97	946.7	**−516.1409**	**−840.0497**	**−662.7313**	**−561.7078**	**−840.0497**	**2**
1783.64	1597.99	2261.31	**−238.2212**	**−229.1473**	**−249.5309**	**−84.3845**	**−249.5309**	**3**
198.83	3250.45	2445.08	**−384.1623**	**−417.6881**	**−152.1706**	**−451.3320**	**−451.3320**	**4**
1494.63	2072.59	2550.51	**−173.2142**	**−171.0475**	**−173.6852**	**−90.6772**	**−173.6852**	**3**
1597.03	1921.52	2126.76	**−190.2958**	**−197.7363**	**−210.8427**	**−91.9974**	**−210.8427**	**3**
1598.93	1921.08	1623.33	**−201.1545**	**−233.5756**	**−247.6003**	**−117.9971**	**−247.6003**	**3**
1243.13	1814.07	3441.07	**−219.8759**	**−200.7782**	**−173.5703**	**−134.8556**	**−219.8759**	**1**
2336.31	2640.26	1599.63	**−375.2066**	**−505.3220**	**−432.9655**	**−264.1726**	**−505.3220**	**2**
354	3300.12	2373.61	**−342.0554**	**−389.9420**	**−159.5456**	**−420.4420**	**−420.4420**	**4**
2144.47	2501.62	591.51	**−391.7323**	**−602.1676**	**−534.2673**	**−353.3480**	**−602.1676**	**2**
426.31	3105.29	2057.8	**−292.8245**	**−350.3599**	**−158.4555**	**−372.8588**	**−372.8588**	**4**
1507.13	1556.89	1954.51	**−215.8337**	**−208.6549**	**−228.7517**	**−97.1952**	**−228.7517**	**3**
343.07	3271.72	2036.94	**−331.3153**	**−401.5729**	**−176.3704**	**−427.3639**	**−427.3639**	**4**
2201.94	3196.22	935.53	**−444.7721**	**−718.8749**	**−585.7638**	**−464.5642**	**−718.8749**	**2**
2232.43	3077.87	1298.87	**−404.5625**	**−617.0008**	**−507.0459**	**−379.8982**	**−617.0008**	**2**
1580.1	1752.07	2463.04	**−200.0040**	**−188.5023**	**−202.9287**	**−82.0465**	**−202.9287**	**3**
1962.4	1594.97	1835.95	**−280.3708**	**−289.5207**	**−305.2451**	**−108.2437**	**−305.2451**	**3**
1495.18	1957.44	3498.02	**−201.1284**	**−185.7029**	**−178.9540**	**−105.8759**	**−201.1284**	**1**
1125.17	1594.39	2937.73	**−229.6958**	**−204.2460**	**−176.7869**	**−136.2208**	**−229.6958**	**1**
24.22	3447.31	2145.01	**−454.8835**	**−518.7193**	**−193.0544**	**−551.2891**	**−551.2891**	**4**
1269.07	1910.72	2701.97	**−182.6951**	**−166.4209**	**−156.1743**	**−104.4475**	**−182.6951**	**1**
1802.07	1725.81	1966.35	**−234.1142**	**−241.7995**	**−260.4340**	**−95.1993**	**−182.6951**	**3**
1817.36	1927.4	2328.79	**−214.3639**	**−219.7679**	**−232.7225**	**−85.5928**	**−232.7225**	**3**
1860.45	1782.88	1875.13	**−242.2520**	**−258.9318**	**−274.4652**	**−274.4652**	**−274.4652**	**3**

对比正确分类后发现，存在一组数据(1494.63，2072.59，2550.51)与正确分类有出入，用上述方法得到结果属于类别3，而正确的分类结果是属于类别1。这可能是基于最小风险贝叶斯分类的分类方法存在误差所导致的。

反过来验证分类结果的正确性。首先修改 MATLAB 循环语句的循环次数与后验概率的输入向量，程序代码如下：

```
for k = 1:29
P1 = -1/2 * (pattern(k,:)' - X1)' * S1_ * (pattern(k,:)' - X1) + log(Pw1) - 1/2 * log(S11);
P2 = -1/2 * (pattern(k,:)' - X2)' * S2_ * (pattern(k,:)' - X2) + log(Pw2) - 1/2 * log(S22);
P3 = -1/2 * (pattern(k,:)' - X3)' * S3_ * (pattern(k,:)' - X3) + log(Pw3) - 1/2 * log(S33);
P4 = -1/2 * (pattern(k,:)' - X4)' * S4_ * (pattern(k,:)' - X4) + log(Pw4) - 1/2 * log(S44);
```

执行程序后 MATLAB 命令窗口将显示结果，摘录其中一部分如下：

```
risk =
 -223.2045   -213.1886   -231.9085   -80.3811
minriskloss =
 -231.9085
w =
3
risk =
 -315.9476   -345.1539   -136.7743   -373.5723
minriskloss =
 -373.5723
w =
4
risk =
 -246.9909   -270.3278   -291.5826   -119.2448
minriskloss =
 -291.5826
w =
3
risk =
 -241.1256   -213.7000   -157.3226   -168.3969
minriskloss =
 -241.1256
w =
1
```

这与训练样本的分类结果是完全吻合的。

2.3.5 结论

以贝叶斯决策为核心内容的统计决策理论是统计模式识别的重要基础，理论上该分类理论就有最优性能，即分类错误或风险在所有分类器中是最小的，常可以作为衡量其他分类器设计方法的优劣标准。

但是该方法明显的局限在于：需要已知类别数以及各类别的先验概率和类条件概率密

度。也就是说，要分两步来解决模式识别问题——先根据训练样本设计分类器，接着对测试样本进行分类。因此，有必要研究直接从测试样本出发设计分类器的其他方法。

习题

（1）什么是最小错误率贝叶斯决策？

（2）什么是最小风险贝叶斯决策？

（3）最小错误率贝叶斯决策与最小风险贝叶斯决策的区别是什么？

第3章

判别函数分类器设计

3.1 判别函数简介

判别函数是统计模式识别中用以对模式进行分类的一种较简单的函数。在特征空间中，通过学习，不同的类别可以得到不同的判别函数，比较不同类别的判别函数值的大小，就可以进行分类。统计模式识别方法把特征空间划分为决策区对模式进行分类，一个模式类同一个或几个决策区相对应。

设有 c 个类别，对于每一个类别 $\omega_i(i=1,2,\cdots,c)$定义一个关于特征向量 $\boldsymbol{X}$ 的单值函数 $g_i(\boldsymbol{X})$：①如果 $\boldsymbol{X}$ 属于第 i 类，那么 $g_i(\boldsymbol{X})>g_j(\boldsymbol{X})(i,j=1,2,\cdots,c,j\neq i)$；②如果 $\boldsymbol{X}$ 在第 i 类和第 j 类的分界面上，那么 $g_i(\boldsymbol{X})=g_j(\boldsymbol{X})(i,j=1,2,\cdots,c,j\neq i)$。

人们已研究出多种求取决策边界的算法，线性判别函数的决策边界是一个超平面方程式，其中的系数可以从已知类别的学习样本集求得。F.罗森布拉特的错误修正训练程序是求取两类线性可分分类器决策边界的早期方法之一。在用线性判别函数不可能对所有学习样本正确分类的情况下，可以规定一个准则函数(例如对学习样本的错分数最少)并用使准则函数达到最优的算法求取决策边界。用线性判别函数的模式分类器也称为线性分类器或线性机，这种分类器计算简单，不要求估计特征向量的类条件概率密度，是一种非参数分类方法。

当用贝叶斯决策理论进行分类器设计时，在一定的假设下也可以得到线性判别函数，这无论对于线性可分或线性不可分的情况都是适用的。在问题比较复杂的情况下可以用多段线性判别函数(见近邻法分类、最小距离分类)或多项式判别函数对模式进行分类。一个二阶的多项式判别函数可以表示为与它相应的决策边界是一个超二次曲面。

本章介绍线性判别函数和非线性判别函数，用以对酒瓶的颜色进行分类，其中实现线性判别函数分类的方法有 LMSE 分类算法和 Fisher 分类，实现非线性判别函数分类的方法有基于核的 Fisher 分类和支持向量机。

3.2 线性判别函数

判别函数分为线性判别函数和非线性判别函数。最简单的判别函数是线性判别函数，它是由所有特征量的线性组合构成的。我们现在对两类问题和多类问题分别进行讨论。

1. 两类问题

对于两类问题，也就是 $\boldsymbol{W}_i=(\omega_1,\omega_2)^{\mathrm{T}}$。

1）二维情况

取二维特征向量 $\boldsymbol{X}=(x_1,x_2)^{\mathrm{T}}$，这种情况下的判别函数 $g(x)=\omega_1x_1+\omega_2x_2+\omega_3$，其中，$\omega_i(i=1,2,3)$为参数；$x_1$ 和 x_2 为坐标值，判别函数 $g(x)$具有以下性质：当 $x\in\omega_1$ 时，$g_i(x)>0$；当 $x\in\omega_2$ 时，$g_i(x)<0$；当 x 不定时，$g_i(x)=0$。这是二维情况下由判别边界分类。

2）n 维情况

对于 n 维情况，现抽取 n 维特征向量：$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$，判别函数为 $g(x)=\boldsymbol{W}_0\boldsymbol{X}+\omega_{n+1}$。其中，$\boldsymbol{W}_0=(\omega_1,\omega_2,\cdots,\omega_n)^{\mathrm{T}}$ 为权向量；$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$ 为模式向量。另外一种表示方法是 $g(x)=\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}$。其中，$\boldsymbol{W}=(\omega_1,\omega_2,\cdots,\omega_n,\omega_{n+1})^{\mathrm{T}}$ 为增值权向量；$\boldsymbol{X}=(x_1,x_2,\cdots,x_n,1)^{\mathrm{T}}$ 为增值模式向量。

在这种情况下，当 $x\in\omega_1$ 时，$g(x)>0$；当 $x\in\omega_2$ 时，$g(x)<0$；$g_1(x)=0$ 为边界，当 $n=2$ 时，边界为一条直线，当 $n=3$ 时，边界为一个平面，当 $n>3$ 时，边界为超平面。

2. 多类问题

对于多类问题，模式有 $\omega_1,\omega_2,\cdots,\omega_M$ 个类别，可以分为下面三种情况。

1）第一种情况

每个模式类与其他模式可用单个判别平面分开，这时 M 个类有 M 个判别函数，且具有性质

$$g_i(x)=\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X} \tag{3-1}$$

式中，$\boldsymbol{W}_i=(\omega_{i1},\omega_{i2},\cdots,\omega_{in+1})^{\mathrm{T}}$ 为第 i 个判别函数的权向量。当 $x\in\omega_i$ 时，$g_i(x)>0$，其他情况下 $g_i(x)<0$，也就是每一个类别可以用单个判别边界与其他类别相分开。

2）第二种情况

每个模式类和其他模式类之间可以用判别平面分开，这样就有$\frac{M(M-1)}{2}$个平面，对于两类问题，$M=2$，则有 1 个判别平面，同理对于三类问题，就有 3 个判别平面。判别函数为

$$g_{ij}(x)=\boldsymbol{W}_{ij}^{\mathrm{T}}\boldsymbol{X} \tag{3-2}$$

式中，$i\neq j$，判别边界为 $g_{ij}(x)=0$，条件为：当 $x\in\omega_i$ 时，$g_{ij}(x)>0$；当 $x\in\omega_j$ 时，$g_{ij}(x)<0$。

3）第三种情况

每类都有一个判别函数，存在 M 个判别函数：$g_k(x)=\boldsymbol{W}_k\boldsymbol{X}(k=1,2,\cdots,M)$，边界为 $g_i(x)=g_j(x)$，条件为：当 $x\in\omega_i$ 时，$g_i(x)$最大；其他情况下 $g_i(x)$小。也就是说，要判别 $\boldsymbol{X}$ 属于哪一个类，先把 $\boldsymbol{X}$ 代入 M 个判别函数，判别函数最大的那个类就是 $\boldsymbol{X}$ 所属类别。

3.3 线性判别函数的实现

对于给定的样本集 $\boldsymbol{X}$,要确定线性判别函数 $g(x)=\boldsymbol{W}^{\mathrm{T}}x+\omega_0$ 的各项系数 $\boldsymbol{W}$ 和 ω_0,可以通过以下步骤来实现:

① 收集一组具有类别标志的样本 $\boldsymbol{X}=\{x_1,x_2,\cdots,x_N\}$;

② 按照需要确定准则函数 J;

③ 用最优化技术求准则函数 J 的极值解 ω^* 和 ω_0^*,从而确定判别函数,完成分类器的设计。

对于未知样本 x,计算 $g(x)$,判断其类别。即对于一个线性判别函数,主要任务是确定线性方程的两个参数,一个是权向量 $\boldsymbol{W}$,另一个是阈值 ω_0。

在计算机中想要实现线性判别函数,可以通过“训练”和“学习”的方式,将已知样本放入到计算机的“训练”程序,经过多次迭代,从而得到准确函数。

下面具体介绍各种分类器的设计。

3.4 基于 LMSE 的分类器设计

3.4.1 LMSE 分类法简介

LMSE 是 Least Mean Square Error 的英文缩写,中文的意思是最小均方误差,常称作 LMSE 算法。

提到 LMSE 分类算法就不能不提感知器算法和自适应算法,因为 LMSE 算法本身就是自适应算法中最常用的方法,而感知器和自适应线性元件在历史上几乎是同时提出的,并且两者在对权值的调整的算法非常相似,它们都是基于纠错学习规则的学习算法。感知器算法存在如下问题:不能推广到一般的前向网络中;函数不是线性可分时,得不出任何结果。而由美国斯坦福大学的 Widrow-Hoff 在研究自适应理论时提出的 LMSE 算法,由于其易实现因而很快得到了广泛应用,成为自适应滤波的标准算法。下面介绍自适应过程。

自适应过程是一个不断逼近目标的过程。它所遵循的途径以数学模型表示,称为自适应算法。通常采用基于梯度的算法,其中 LMSE 算法尤为常用。自适应算法可以用硬件(处理电路)或软件(程序控制)两种办法实现。前者依据算法的数学模型设计电路,后者则将算法的数学模型编制成程序并用计算机实现。算法有很多种,选择算法很重要,它决定了处理系统的性能质量和可行性。

自适应均衡器的原理就是按照某种准则和算法对其系数进行调整,最终使自适应均衡器的代价(目标)函数最小化,达到最佳均衡的目的。而各种调整系数的算法就称为自适应算法,自适应算法是根据某个最优准则来设计的。最常用的自适应算法有逼零算法、最陡下降算法、LMSE 算法、RLS 算法以及各种盲均衡算法等。

自适应算法所采用的最优准则有最小均方误差准则、最小二乘准则、最大信噪比准则和统计检测准则等，其中最小均方误差准则和最小二乘准则是目前最为流行的自适应算法准则。LMSE 算法和 RLS 算法由于采用的最优准则不同，因此这两种算法在性能、复杂度等方面均有许多差别。

一种算法性能的好坏可以通过几个常用的指标来衡量，例如收敛速度——通常用算法达到稳定状态(即与最优值的接近程度达到一定值)的迭代次数表示；误调比——实际均方误差相对于算法的最小均方误差的平均偏差；运算复杂度——完成一次完整迭代所需的运算次数；跟踪性能——对信道时变统计特性的自适应能力。

3.4.2 LMSE 算法原理

LMSE 算法是针对准则函数引进最小均方误差这一条件而建立起来的。这种算法的主要特点是在训练过程中判定训练集是否线性可分，从而可对结果的收敛性做出判断。

LMSE 算法属于监督学习的类型，而且是“模型无关”的，它是通过最小化输出和期望目标值之间的偏差来实现的。

LMSE 算法属于自适应算法中常用的算法，它不同于 C 均值算法和 ISODATA 算法，后两种属于基于距离度量的算法，直观且容易理解。LMSE 算法通过调整权值函数求出判别函数，进而将待测样本代入判别函数求值，最终做出判定，得出答案。

1. 准则函数

LMSE 算法以最小均方差作为准则，因均方差为

$$E\{[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}]^2\} \tag{3-3}$$

因而准则函数为

$$J(\boldsymbol{W}_i,\boldsymbol{X})=\frac{1}{2}E\{[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}]^2\} \tag{3-4}$$

准则函数在 $r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}=0$ 时取得最小值。准则函数对 $\boldsymbol{W}_i$ 的偏导数为

$$\frac{\partial J}{\partial \boldsymbol{W}_i}=E\{-\boldsymbol{X}[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{X}]\} \tag{3-5}$$

2. 迭代方程

将式(3-5)代入迭代方程，得到

$$\boldsymbol{W}_i(k+1)=\boldsymbol{W}_i(k)+\alpha_k\boldsymbol{X}(k)[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}(k)\boldsymbol{X}(k)] \tag{3-6}$$

对于多类问题来说，M 类问题应该有 M 个权函数方程，而对于每一个权函数方程来说，如 $\boldsymbol{X}(k)\in\omega_i$，则

$$r_i[\boldsymbol{X}(k)]=1,\quad i=1,2,\cdots,M \tag{3-7}$$

否则

$$r_i[\boldsymbol{X}(k)]=0,\quad i=1,2,\cdots,M \tag{3-8}$$

3.4.3 LMSE 算法步骤

(1) 设各个权向量的初始值为 0，即 $\boldsymbol{W}_0(0)=\boldsymbol{W}_1(0)=\boldsymbol{W}_2(0)=\cdots=\boldsymbol{W}_M(0)=0$。

(2) 输入第 k 次样本 $\boldsymbol{X}(k)$,计算 $d_i(k)=\boldsymbol{W}_i^{\mathrm{T}}(k)\boldsymbol{X}(k)$。

(3) 确定期望输出函数值:若 $\boldsymbol{X}(k)\in\omega_i$,则 $r_i[\boldsymbol{X}(k)]=1$,否则 $r_i[\boldsymbol{X}(k)]=0$。

(4) 计算迭代方程:$\boldsymbol{W}_i(k+1)=\boldsymbol{W}_i(k)+\alpha_k\boldsymbol{X}(k)[r_i(\boldsymbol{X})-\boldsymbol{W}_i^{\mathrm{T}}(k)\boldsymbol{X}(k)]$,其中 $\alpha_k=\frac{1}{k}$。

(5) 循环执行步骤(2),直到满足条件:属于 ω_i 类的所有样本都满足不等式 $d_i(\boldsymbol{X})>d_j(\boldsymbol{X}),\forall j\neq i$。

3.4.4 LMSE 算法的 MATLAB 实现

1. 首先给定四类样本,各样本的特征向量经过增 1

程序如下:

```
pattern = struct('feature',[])
p1 = [864.45   1647.31  2665.9;
      877.88   2031.66  3071.18;
      1418.79  1775.89  2772.9;
      1449.58  1641.58  3405.12;
      864.45   1647.31  2665.9;
      877.88   2031.66  3071.18;
      1418.79  1775.89  2772.9;
      1449.58  1641.58  3405.12;
      1418.79  1775.89  2772.9;
      1449.58  1641.58  3405.12;]
pattern(1).feature = p1'
pattern(1).feature(4,:) = 1
```

pattern(1).feature 实际的矩阵形式如下:

```
p1 =
  1.0e + 03 *
    0.8645    1.6473    2.6659
    0.8779    2.0317    3.0712
    1.4188    1.7759    2.7729
    1.4496    1.6416    3.4051
    0.8645    1.6473    2.6659
    0.8779    2.0317    3.0712
    1.4188    1.7759    2.7729
    1.4496    1.6416    3.4051
    1.4188    1.7759    2.7729
    1.4496    1.6416    3.4051
```

之后的三类,程序如下:

```
p2 = [2352.12  2557.04  1411.53;
2297.28  3340.14  535.62;
2092.62  3177.21  584.32;
2205.36  3243.74  1202.69;
2949.16  3244.44  662.42;
2802.88  3017.11  1984.98;
2063.54  3199.76  1257.21;
```

```
2949.16  3244.44  662.42;
2802.88  3017.11  1984.98;
2063.54  3199.76  1257.21;]
pattern(2).feature = p2'
pattern(2).feature(4,:) = 1
p3 = [1739.94  1675.15  2395.96;
1756.77  1652  1514.98;
1803.58  1583.12  2163.05;
1571.17  1731.04  1735.33;
1845.59  1918.81  2226.49;
1692.62  1867.5  2108.97;
1680.67  1575.78  1725.1;
1651.52  1713.28  1570.38;
1680.67  1575.78  1725.1;
1651.52  1713.28  1570.38;]
pattern(3).feature = p3'
pattern(3).feature(4,:) = 1
p4 = [373.3  3087.05  2429.47;
222.85  3059.54  2002.33;
401.3  3259.94  2150.98;
363.34  3477.95  2462.86;
104.8  3389.83  2421.83;
499.85  3305.75  2196.22;
172.78  3084.49  2328.65;
341.59  3076.62  2438.63;
291.02  3095.68  2088.95;
237.63  3077.78  2251.96;]
pattern(4).feature = p4'
pattern(4).feature(4,:) = 1
```

2. 设权值向量的初始值均为 0

初始化权值程序代码如下：

```
w = zeros(4,4);   % 初始化权值
```

MATLAB 程序运行结果如下：

```
w =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

3. 计算 di(k)

程序代码如下：

```
for k = 1:4
                m = pattern(i).feature(:,j)
                m = m/norm(m)
                d(k) = w(:,k)' * m      % 计算 d
```

MATLAB程序运行结果如下。

第一次循环结果：

```
m =
  1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0
m =
  1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0     0
m =
  1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0     0     0
m =
    1.0e+03 *
    0.8645
    1.6473
    2.6659
    0.0010
m =
    0.2659
    0.5067
    0.8201
    0.0003
d =
    0     0     0     0
```

最后一次运行结果：

```
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.3263    0.3348    0.1432
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.1397    0.3348    0.1432
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.1397    0.1217    0.1432
m =
  1.0e+03 *
    0.2376
    3.0778
    2.2520
    0.0010
m =
    0.0622
    0.8055
    0.5894
    0.0003
d =
    0.2745    0.1397    0.1217    0.4591
```

4. 调整权值

程序代码如下：

```
for k = 1:4
                if k~ = i
                    if d(i)<= d(k)        % d(i)不是最大,则继续迭代
                        flag = 1;
                    end
                end
            end
            % 调整权值
            for k = 1:4
                w(:,k) = w(:,k) + m * (r(k) - d(k))/num
```

MATLAB 程序运行结果如下：

```
w =
   - 0.1204    0.4917    0.4122   - 0.4088
   - 0.1820    0.3990   - 0.0421    0.4791
     0.7271   - 0.3601    0.2204    0.1674
     0.0001    0.0000    0.0002    0.0000
```

5. 通过判别函数将待分类数据分类

调用 function 函数,将待测数据分类。因为调用该函数一次只能判别一个样品的类别,所以循环 30 次才能完成分类,程序代码如下：

```
for k = 1:30
    sample = sampletotall(:,k)
    y = lmseclassify(sample)
    x = sample(1)
    yy = sample(2)
    z = sample(3)
    ac(k) = y
```

运行 MATLAB 程序,最终的分类结果如下：

```
ac =
1 ～ 15 列
3    3    1    3    4    2    2    3    4    1    3    2    1    2
4
16 ～ 30 列
2    4    3    4    2    2    1    3    1    1    4    1    3    3
3
```

将该分类结果与原始分类结果对比,对照表如表 3-1 所示。

表 3-1 LMSE 分类结果与原始分类结果对照表

序　　号	A	B	C	原始分类结果	LMSE 分类结果
1	1702.8	1639.79	2068.74	3	3
2	1877.93	1860.96	1975.3	3	3
3	867.81	2334.68	2535.1	1	1
4	1831.49	1713.11	1604.68	3	3
5	460.69	3274.77	2172.99	4	4
6	2374.98	3346.98	975.31	2	2
7	2271.89	3482.97	946.7	2	2
8	1783.64	1597.99	2261.31	3	3
9	198.83	3250.45	2445.08	4	4
10	1494.63	2072.59	2550.51	1	1
11	1597.03	1921.52	2126.76	3	3
12	1598.93	1921.08	1623.33	3	2
13	1243.13	1814.07	3441.07	1	1
14	2336.31	2640.26	1599.63	2	2
15	354	3300.12	2373.61	4	4
16	2144.47	2501.62	591.51	2	2
17	426.31	3105.29	2057.8	4	4
18	1507.13	1556.89	1954.51	3	3
19	343.07	3271.72	2036.94	4	4
20	2201.94	3196.22	935.53	2	2
21	2232.43	3077.87	1298.87	2	2
22	1580.1	1752.07	2463.04	3	1
23	1962.4	1594.97	1835.95	3	3
24	1495.18	1957.44	3498.02	1	1
25	1125.17	1594.39	2937.73	1	1
26	24.22	3447.31	2145.01	4	4
27	1269.07	1910.72	2701.97	1	1
28	1802.07	1725.81	1966.35	3	3
29	1817.36	1927.4	2328.79	3	3
30	1860.45	1782.88	1875.13	3	3

结果分析：从表 3-1 中可以看出有 2 个分类结果是错的，正确率为 93.3%。

6. 用三维效果图将结果直观显示

将分好类的数据用三维图像的形式直观显示出来，程序代码如下：

```
axis([0 3500 0 3500 0 3500])
 if y == 1
        plot3(x,yy,z,'g * ');        % 第一类表示为绿色
   elseif y == 2
        plot3(x,yy,z,'r * ')         % 第二类表示为红色
   elseif y == 3
```

```
        plot3(x,yy,z,'b*')          %第三类表示为蓝色
      elseif y==4
        plot3(x,yy,z,'y*')          %第四类表示为黄色
end
hold on
```

运行 MATLAB 程序，三维效果图如图 3-1 所示。

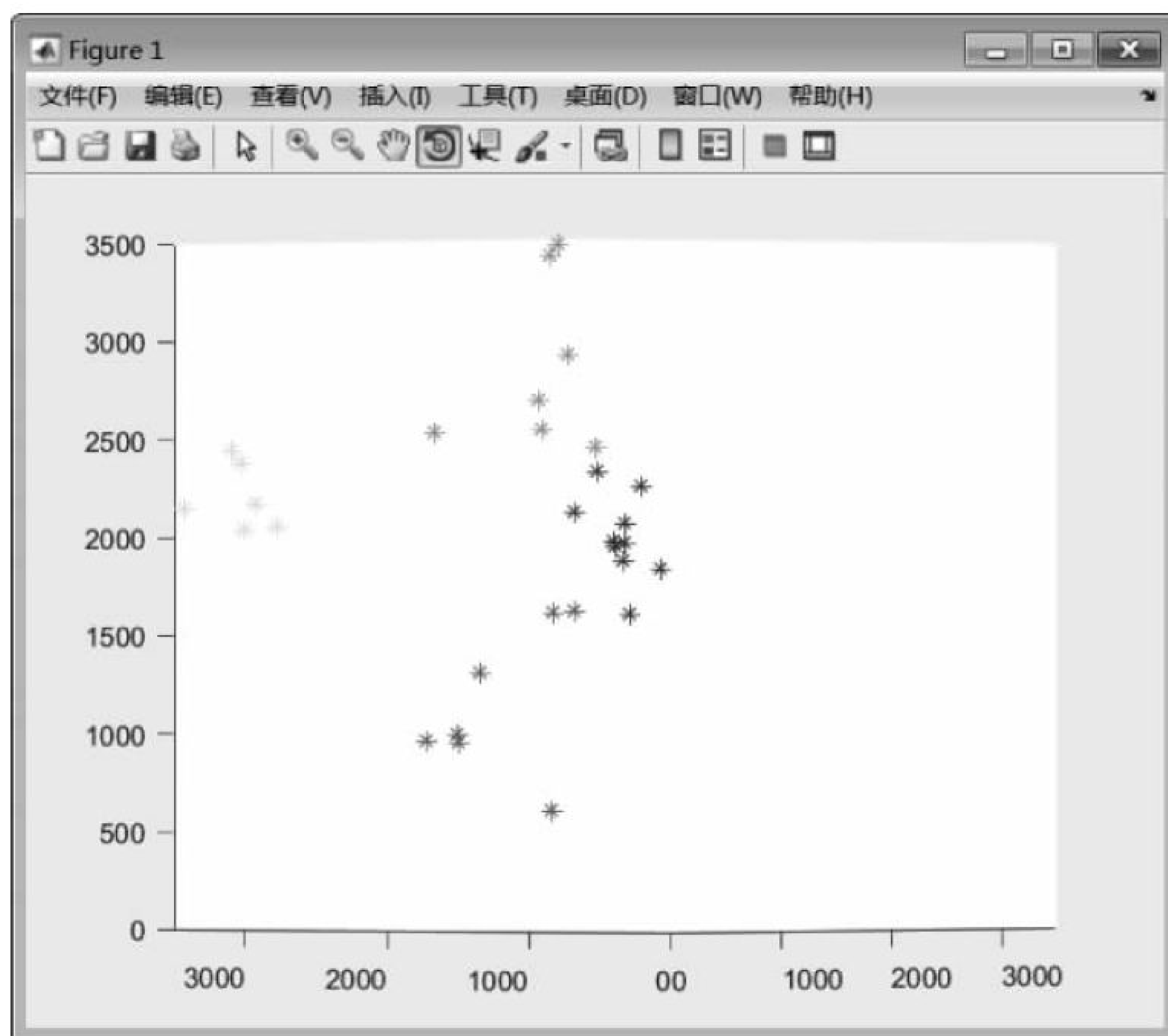


图 3-1 LMSE 算法分类结果三维效果图

7. 完整的 MATLAB 程序

完整的 MATLAB 程序代码如下：

```
%主程序
%输入数据
clear all
close all
clc
s1 = xlsread('C:\Users\Administrator\Desktop\ln.xls')   %从 Excel 表格直接读入数据
samp = s1(30:59,1:3)                                     %后 30 个数据
sampletotall = samp'                                     %转置
ac = []                                                  %定义分类矩阵
for k = 1:30
    sample = sampletotall(:,k)
    y = lmseclassify(sample)                             %调用 function 函数
    x = sample(1)
```

```
    yy = sample(2)
    z = sample(3)
    ac(k) = y
axis([0 3500 0 3500 0 3500])
    if y == 1
        plot3(x,yy,z,'g*')              %第一类用绿色表示
    elseif y == 2
        plot3(x,yy,z,'r*')              %第二类用红色表示
    elseif y == 3
        plot3(x,yy,z,'b*')              %第三类用蓝色表示
    elseif y == 4
        plot3(x,yy,z,'y*')              %第四类用黄色表示
    end
    hold on
end

%利用LMSE算法进行分类的function函数
function y = lmseclassify(sample)
 clc;
 pattern = struct('feature',[])
p1 = [
    864.45    1647.31   2665.9;
    877.88    2031.66   3071.18;
    1418.79   1775.89   2772.9;
    1449.58   1641.58   3405.12;
    864.45    1647.31   2665.9;
    877.88    2031.66   3071.18;
    1418.79   1775.89   2772.9;
    1449.58   1641.58   3405.12;
    1418.79   1775.89   2772.9;
    1449.58   1641.58   3405.12;]
pattern(1).feature = p1'
pattern(1).feature(4,:) = 1
p2 = [2352.12   2557.04   1411.53;
     2297.28   3340.14   535.62;
     2092.62   3177.21   584.32;
     2205.36   3243.74   1202.69;
     2949.16   3244.44   662.42;
     2802.88   3017.11   1984.98;
     2063.54   3199.76   1257.21;
     2949.16   3244.44   662.42;
     2802.88   3017.11   1984.98;
     2063.54   3199.76   1257.21;]
pattern(2).feature = p2'
pattern(2).feature(4,:) = 1
p3 = [1739.94   1675.15   2395.96;
     1756.77   1652      1514.98;
     1803.58   1583.12   2163.05;
     1571.17   1731.04   1735.33;
     1845.59   1918.81   2226.49;
```

```
        1692.62  1867.5   2108.97;
        1680.67  1575.78  1725.1;
        1651.52  1713.28  1570.38;
        1680.67  1575.78  1725.1;
        1651.52  1713.28  1570.38;]
pattern(3).feature = p3'
pattern(3).feature(4,:) = 1
p4 = [373.3    3087.05  2429.47;
      222.85   3059.54  2002.33;
      401.3    3259.94  2150.98;
      363.34   3477.95  2462.86;
      104.8    3389.83  2421.83;
      499.85   3305.75  2196.22;
      172.78   3084.49  2328.65;
      341.59   3076.62  2438.63;
      291.02   3095.68  2088.95;
      237.63   3077.78  2251.96;]
pattern(4).feature = p4'
pattern(4).feature(4,:) = 1

w = zeros(4,4)        % 初始化权值
 flag = 1;
 num = 0;
 num1 = 0;
 d = []
 m = []
 r = [];
s = xlsread('C:\Users\Administrator\Desktop\ln.xls')
 while flag
     flag = 0;
     num1 = num1 + 1
     for j = 1:10
          for i = 1:4
                num = num + 1;
                r = [0 0 0 0];
                r(i) = 1;
                for k = 1:4
                     m = pattern(i).feature(:,j)
                     m = m/norm(m)
                     d(k) = w(:,k)' * m;         % 计算 d
                end
                for k = 1:4
                     if k~ = i
                           if d(i)<= d(k)     % d(i)不是最大,则继续迭代
                                flag = 1;
                           end
                     end
                end
                % 调整权值
                num;
```

```
                num1;
                for k = 1:4
                    w(:,k) = w(:,k) + m * (r(k) - d(k))/num;
                end
            end
        end
        if num1 > 200                           % 超过迭代次数则退出
            flag = 0;
        end
    end
    sample(4) = 1
     h = [];
    for k = 1:4
        h(k) = w(:,k)' * sample;                % 计算判别函数
    end
    [maxval,maxpos] = max(h);
    y = maxpos;
```

3.4.5 结论

学习样本的维数问题：

因为样本类别不均匀(第一类 4 个样本，第二类 8 个样本，第三类 9 个样本，第四类 10 个样本)，程序不运行，后来将数据重复添加进去，保证了程序的正常运行。程序相关部分代码如下：

```
        for j = 1:10
                ……
                  for k = 1:4
                        m = pattern(i).feature(:,j)
                        d(k) = w(:,k)' * m      % 计算 d
                  end
                ……
        end
p1 = [864.45   1647.31   2665.9;877.88   2031.66   3071.18;1418.79   1775.89   2772.9;1449.58
      1641.58   3405.12;864.45   1647.31   2665.9;877.88   2031.66   3071.18;1418.79   1775.89
      2772.9;1449.58   1641.58   3405.12;1418.79   1775.89   2772.9;1449.58   1641.58
      3405.12;]
```

注意：其中 864.45 1647.31 2665.9；877.88 2031.66 3071.18；1418.79 1775.89 2772.9；1449.58 1641.58 3405.12；1418.79 1775.89 2772.9；1449.58 1641.58 3405.12 是重复添加的样本数据，目的是凑够 10 个数据以便程序能进行循环。

3.5 基于 Fisher 的分类器设计

3.5.1 Fisher 判别法简介

Fisher 判别法是 1936 年由 R. A. Fisher 首先提出的。Fisher 判别法是一种线性判别法，线性判别又称线性准则。与线性准则相对应的还有非线性准则，其中一些在变换条件下可以转化为线性准则，因此对应于 d 维特征空间，线性判别函数虽然最简单，但是在应用上却具有普遍意义，便于对分类问题的理解与描述。

基于线性判别函数的线性分类方法，虽然使用有限样本集合来构造，从严格意义上来讲属于统计分类方法。也就是说，对于线性分类器的检验，应建立在样本扩充的条件下，以基于概率的尺度来评价才是有效的评价。尽管线性分类器的设计在满足统计学的评价下并不严格与完美，但是由于其具有的简单性与实用性，在分类器设计中还是获得了广泛的应用。

3.5.2 Fisher 判别法的原理

设计线性分类器首先要确定准则函数，然后再利用训练样本集确定该分类器的参数，以求使所确定的准则达到最佳。在使用线性分类器时，样本的分类由其判别函数值决定，而每个样本的判别函数值是其各分量的线性加权和再加上一个阈值 y_0。

如果只考虑各分量的线性加权和，则它是各样本向量与向量 $\boldsymbol{w}$ 的向量点积。如果向量 $\boldsymbol{w}$ 的幅度为单位长度，则线性加权和又可被看作各样本向量在向量 $\boldsymbol{w}$ 上的投影。

Fisher 判别法的基本原理是，对于 d 维空间的样本，投影到一维坐标上，样本特征将混杂在一起，难以区分。Fisher 判别法的目的，就是要找到一个最合适的投影轴 $\boldsymbol{w}$，使两类样本在该轴上投影的交叠部分最少，从而使分类效果为最佳。如何寻找一个投影方向，使得样本集合在该投影方向上最易区分，这就是 Fisher 判别法所要解决的问题。Fisher 投影原理如图 3-2 所示。

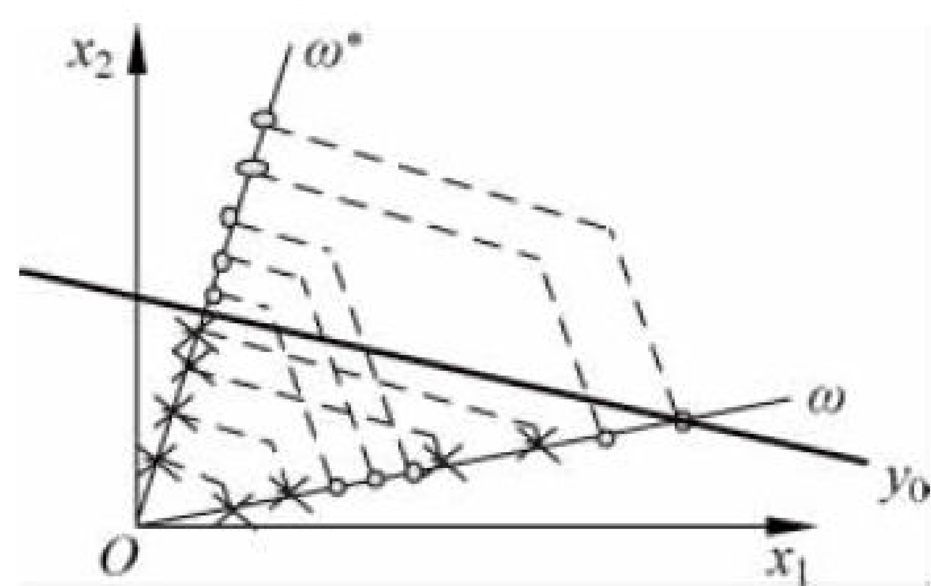


图 3-2 Fisher 投影原理图

Fisher 准则函数的基本思路：即向量 $\boldsymbol{w}$ 的方向选择应能使两类样本投影的均值之差尽可能大些，而使类内样本的离散程度尽可能小。

3.5.3 Fisher 分类器设计

已知 N 个 d 维样本数据集合 $\chi=\{x_1,x_2,\cdots,x_N\}$，其中类别为 $\omega_i(i=1,2)$，样本容量为 N_i，其子集为 x_i，以投影坐标向量 $\boldsymbol{w}$ 与原特征向量 $\boldsymbol{x}$ 作数量积，可得投影表达式为 $y_n=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n(n=1,2,\cdots,N)$。

相应地，y_n 也为两个子集 y_1 和 y_2。如果只考虑投影向量 $\boldsymbol{w}$ 的方向，不考虑其长度，即

默认其长度为单位1,则 y_n 即为 $\boldsymbol{x}_n$ 在 $\boldsymbol{w}$ 方向上的投影。Fisher 准则的目的就是寻找最优投影方向,使得 $\boldsymbol{w}$ 为最好的投影向量 $\boldsymbol{w}^*$。

样本在 d 维特征空间的一些描述量如下。

(1) 各类样本均值向量 $\boldsymbol{m}_i$

$$\boldsymbol{m}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{x}_j,\quad i=1,2 \tag{3-9}$$

(2) 样本类内离散度矩阵 $\boldsymbol{S}_i$ 与总类内离散度矩阵 $\boldsymbol{S}_w$

$$\boldsymbol{S}_i = \sum_{j=1}^{N_i}(\boldsymbol{x}_j-\boldsymbol{m}_i)(\boldsymbol{x}_j-\boldsymbol{m}_i)^{\mathrm{T}},\quad i=1,2 \tag{3-10}$$

$$\boldsymbol{S}_w = \boldsymbol{S}_1+\boldsymbol{S}_2 \tag{3-11}$$

(3) 样本类间离散度矩阵 $\boldsymbol{S}_{\mathrm{b}}$

$$\boldsymbol{S}_{\mathrm{b}} = (\boldsymbol{m}_1-\boldsymbol{m}_2)(\boldsymbol{m}_1-\boldsymbol{m}_2)^{\mathrm{T}} \tag{3-12}$$

如果在一维上投影,则有各类样本均值向量 $\bar{\boldsymbol{m}}_i$

$$\bar{\boldsymbol{m}}_i = \frac{i}{N_i}\sum_{j=1}^{N_i}y_j,\quad i=1,2 \tag{3-13}$$

样本类内离散度矩阵 $\bar{\boldsymbol{S}}_i$ 与总类内离散度矩阵 $\bar{\boldsymbol{S}}_w$

$$\bar{\boldsymbol{S}}_i = \sum_{j=1}^{N_i}(y_j-\bar{\boldsymbol{m}}_i)^2,\quad i=1,2 \tag{3-14}$$

$$\bar{\boldsymbol{S}}_w = \bar{\boldsymbol{S}}_1+\bar{\boldsymbol{S}}_2 \tag{3-15}$$

Fisher 准则函数的定义原则为,希望投影后,在一维空间中样本类别区分清晰,即两类样本的距离越大越好,也就是均值之差($\bar{\boldsymbol{m}}_1-\bar{\boldsymbol{m}}_2$)越大越好;各类样本内部密集,即类内离散度 $\bar{\boldsymbol{S}}_w=\bar{\boldsymbol{S}}_1+\bar{\boldsymbol{S}}_2$ 越小越好。根据上述两条原则,构造 Fisher 准则函数

$$J_{\mathrm{F}}(\boldsymbol{w}) = \frac{(\bar{\boldsymbol{m}}_1-\bar{\boldsymbol{m}}_2)^2}{\bar{\boldsymbol{S}}_1+\bar{\boldsymbol{S}}_2} \tag{3-16}$$

使得 $J_{\mathrm{F}}(\boldsymbol{w})$ 为最大值的 $\boldsymbol{w}$ 即为要求的投影向量 $\boldsymbol{w}^*$。

式(3-16)称为 Fisher 准则函数,需进一步转化为 $\boldsymbol{w}$ 的显函数,为此要对 $\bar{\boldsymbol{m}}_1$, $\bar{\boldsymbol{m}}_2$ 等项进一步演化。由于

$$\bar{\boldsymbol{m}}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}y_j = \frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_j = \boldsymbol{w}^{\mathrm{T}}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{x}_j\right) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_i \tag{3-17}$$

则有

$$(\bar{\boldsymbol{m}}_1-\bar{\boldsymbol{m}}_2)^2 = (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_1-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_2)^2 = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{m}_1-\boldsymbol{m}_2)(\boldsymbol{m}_1-\boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}\boldsymbol{w} \tag{3-18}$$

其中 $\boldsymbol{S}_{\mathrm{b}}=(\boldsymbol{m}_1-\boldsymbol{m}_2)(\boldsymbol{m}_1-\boldsymbol{m}_2)^{\mathrm{T}}$ 为类间离散矩阵。再由类内离散度

$$\bar{\boldsymbol{S}}_i = \sum_{j=1}^{N_i}(y_j-\bar{\boldsymbol{m}}_i)^2 = \sum_{j=1}^{N_i}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_j-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_i)^2 = \boldsymbol{w}^{\mathrm{T}}\left[\sum_{j=1}^{N_i}(\boldsymbol{x}_j-\boldsymbol{m}_i)(\boldsymbol{x}_j-\boldsymbol{m}_i)^{\mathrm{T}}\right]\boldsymbol{w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_i\boldsymbol{w} \tag{3-19}$$

其中 $\boldsymbol{S}_i = \sum_{j=1}^{N_i}(\boldsymbol{x}_j-\boldsymbol{m}_i)(\boldsymbol{x}_j-\boldsymbol{m}_i)^{\mathrm{T}}$。

所以总类内离散度为

$$\bar{\boldsymbol{S}}_w = \bar{\boldsymbol{S}}_1+\bar{\boldsymbol{S}}_2 = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{S}_1+\boldsymbol{S}_2)\boldsymbol{w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w} \tag{3-20}$$

将式(3-18)与式(3-20)代入式(3-16),得到 Fisher 准则函数对于变量 $\boldsymbol{w}$ 的显式函数为

$$J_F(\boldsymbol{w}) = \frac{\boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w}} \tag{3-21}$$

对 $\boldsymbol{x}_n$ 的分量作线性组合 $y_n=\boldsymbol{w}^T\boldsymbol{x}_n(n=1,2,\cdots,N)$,从几何意义上看,$\|\boldsymbol{w}\|=1$,则每个 y_n 就是相对应的 $\boldsymbol{x}_n$ 到方向为 $\boldsymbol{w}$ 的直线上的投影。$\boldsymbol{w}$ 的方向不同,将使样本投影后的可分离程序不同,从而直接影响识别效果。

求解 Fisher 准则函数的条件极值,即可解得使 $J_F(\boldsymbol{w})$ 为极值的 $\boldsymbol{w}^*$。对求取其极大值时的 $\boldsymbol{w}^*$,可以采用拉格朗日乘子算法解决,令分母非 0,即 $\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w}=c\neq 0$。

构造拉格朗日函数

$$L(\boldsymbol{w},\lambda) = \boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w} - \lambda(\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w} - c) \tag{3-22}$$

对 $\boldsymbol{w}$ 求偏导,并令其为 0,即

$$\frac{\partial L(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}} = \boldsymbol{S}_b\boldsymbol{w} - \lambda\boldsymbol{S}_w\boldsymbol{w} = 0 \tag{3-23}$$

得到

$$\boldsymbol{S}_b\boldsymbol{w}^* = \lambda\boldsymbol{S}_w\boldsymbol{w}^* \tag{3-24}$$

由于 $\boldsymbol{S}_w$ 非奇异,两边左乘 $\boldsymbol{S}_w^{-1}$,得到 $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b\boldsymbol{w}^*=\lambda\boldsymbol{w}^*$,该式为矩阵 $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ 的特征值问题。其中,拉格朗日算子 λ 为矩阵 $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ 的特征值;$\boldsymbol{w}^*$ 即对应于特征值 λ 的特征向量,即最佳投影的坐标向量。

矩阵特征值的问题有标准的求解方法。在此给出一种直接求解方法,不求特征值而直接得到最优解 $\boldsymbol{w}^*$。

由于

$$\boldsymbol{S}_b = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \tag{3-25}$$

所以

$$\boldsymbol{S}_b\boldsymbol{w}^* = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T\boldsymbol{w}^* = (\boldsymbol{m}_1 - \boldsymbol{m}_2)R$$

式中,$R=(\boldsymbol{m}_1-\boldsymbol{m}_2)^T\boldsymbol{w}^*$ 为限定标量。进而,由于

$$\lambda\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}\boldsymbol{S}_b\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}(\boldsymbol{S}_b\boldsymbol{w}^*) = \boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)R \tag{3-26}$$

得到

$$\boldsymbol{w}^* = \frac{R}{\lambda}\boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \tag{3-27}$$

忽略比例因子 R/λ,得到最优解 $\boldsymbol{w}^*=\boldsymbol{S}_w^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2)$。因此,使得 $J_F(\boldsymbol{w})$ 取极大值时的 $\boldsymbol{w}$ 即为 d 维空间到一维空间的最佳投影方向

$$\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \tag{3-28}$$

向量 $\boldsymbol{w}^*$ 就是使 Fisher 准则函数 $J_F(\boldsymbol{w})$ 达到极大值的解,也就是按 Fisher 准则将 d 维 X 空间投影到一维 Y 空间的最佳投影方向,$\boldsymbol{w}^*$ 的各分量值是对原 d 维特征向量求加权和的权值。

由式(3-28)表示的最佳投影方向是容易理解的,因为其中的$(\boldsymbol{m}_1-\boldsymbol{m}_2)$项是一向量,对与$(\boldsymbol{m}_1-\boldsymbol{m}_2)$平行的向量投影可使两均值点的距离最远。

但是如何使类间分得较开,同时又使类内密集程度较高这样一个综合指标来看,则需根据两类样本的分布离散程度对投影方向作相应的调整,这就体现在对$(\boldsymbol{m}_1-\boldsymbol{m}_2)$向量按 $\boldsymbol{S}_w^{-1}$ 作一线性变换,从而使 Fisher 准则函数达到极值点。

以上讨论了线性判别函数加权向量 $\boldsymbol{w}$ 的确定方法，并讨论了使 Fisher 准则函数极大的 d 维向量 $\boldsymbol{w}^*$ 的计算方法。由 Fisher 判别函数得到了最佳一维投影后，还需确定一个阈值点 y_0，一般可采用以下几种方法确定 y_0，即

$$y_0 = \frac{\bar{\boldsymbol{m}}_1 + \bar{\boldsymbol{m}}_2}{2} \tag{3-29}$$

$$y_0 = \frac{N_1\bar{\boldsymbol{m}}_1 + N_2\bar{\boldsymbol{m}}_2}{N_1 + N_2} \tag{3-30}$$

$$y_0 = \frac{\bar{\boldsymbol{m}}_1 + \bar{\boldsymbol{m}}_2}{2} + \frac{\ln(P(\omega_1)/P(\omega_2))}{N_1 + N_2 - 2} \tag{3-31}$$

式(3-29)是根据两类样本均值之间的平均距离来确定阈值点的。式(3-30)既考虑了样本均值之间的平均距离，又考虑了两类样本的容量大小作阈值位置的偏移修正。式(3-31)既使用了先验概率 $P(\omega_i)$，又考虑了两类样本的容量大小作阈值位置的偏移修正，目的都是使得分类误差尽可能小。

为了确定具体的分界面，还要指定线性方程的常数项。实际工作中可以采用对 y_0 进行逐次修正的方式，选择不同的 y_0 值，计算其对训练样本集的错误率，找到错误率较小的 y_0 值。

对于任意未知类别的样本 $\boldsymbol{x}$，计算它的投影点 $y=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$，决策规则为

$$\begin{cases} y > y_0, & \boldsymbol{x} \in \omega_1 \\ y < y_0, & \boldsymbol{x} \in \omega_2 \end{cases} \tag{3-32}$$

3.5.4 Fisher 算法的 MATLAB 实现

1. 流程图

根据上面所介绍的 Fisher 判别函数，可得出如图 3-3 所示的流程图。

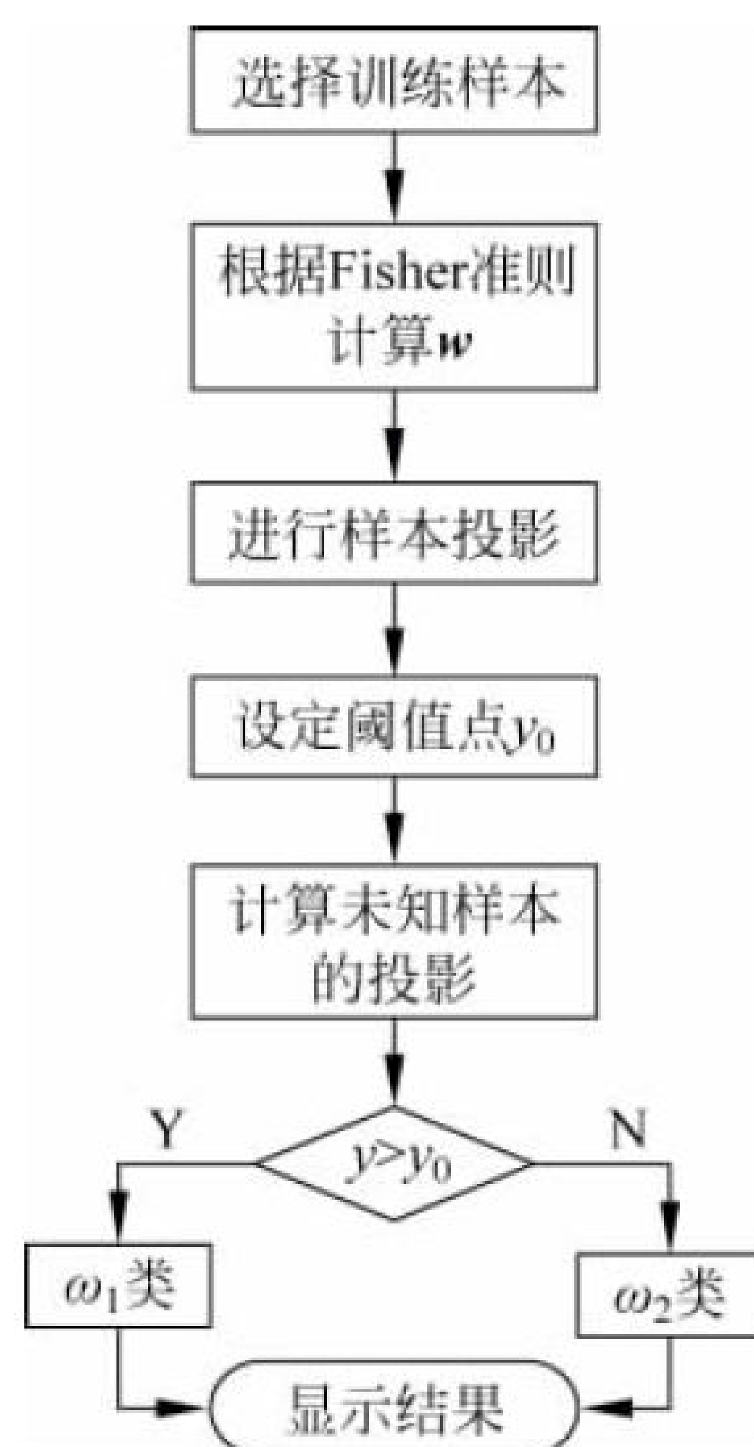


图 3-3 Fisher 分类器设计流程图

2. 样本均值

利用 MATLAB 程序得到训练样本均值,程序代码如下:

```
clear,close all;
N = 29;                                         % N 为训练样本总个数
X = [1495.18    1957.44    3498.02              % X 为训练样本
     1125.17    1594.39    2937.73
     1269.07    1910.72    2701.97
     ……
…… ]
m1 = mean(X(1:11,:));                           % 求得第一类样本均值
m2 = mean(X(12:29,:));                          % 求得第二类样本均值
```

3. 投影向量

Fisher 准则的目的就是寻找最优投影方向,使得 $\boldsymbol{w}$ 为最好的投影向量 $\boldsymbol{w}^*$。

利用如下 MATLAB 程序求得最佳投影向量:

```
S1 = 0;S2 = 0;                                       % 初始化类离散度
for i = 1:11
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';        % 求得第一类的类内离散度
end
for i = 12:29
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';        % 求得第二类的类内离散度
end
Sw = S1 + S2;                                        % 求得总类内离散度
W = inv(Sw) * (m1 - m2);                             % 求得最佳投影方向
```

4. 阈值点

本设计器采用 $y_0 = \boldsymbol{w}^*(\boldsymbol{m}_1 + \boldsymbol{m}_2)^{\mathrm{T}}/2$ 来确定阈值点,由于该式既考虑了样本均值之间的平均距离,又考虑了两类样本的容量大小作阈值位置的偏移修正,因此采用它可以使得分类误差尽可能小。

5. 输出分类结果

对于任意未知类别的样本 $\boldsymbol{x}$,计算它的投影点 $y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$,决策规则为:当 $y > y_0$ 时,$\boldsymbol{x} \in \omega_1$;当 $y < y_0$ 时,$\boldsymbol{x} \in \omega_2$。

输出分类结果的 MATLAB 程序如下:

```
for i = 1:22
y = W * x(i,:)'                   % 确定投影点
```

```
if y>y0                  % 当 y>y0 时,测试样本属于第一类
  disp('一')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r+','MarkerSize',6,'LineWidth',2)
else
  disp('二')             % 当 y<y0 时,测试样本属于第二类
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b+','MarkerSize',6,'LineWidth',2)
end
end
```

3.5.5 识别待测样本类别

本节内容以兑酒为例。不同类型的酒是由多种成分按不同的比例构成的,兑酒时需要三种原料(X,Y,Z),现在已测出不同酒中三种原料的含量,需要判定它属于四种类型中的哪一种。样本中,前 29 组数据用于学习,后 22 组用于识别。

1. 选择分类方法

由于 Fisher 分类法一次只能将样本分成两类,因此,首先要将样本分成两大类,即一类、二类,然后再继续往下分,将其分成 1、2、3、4 类。分类流程图如图 3-4 所示。

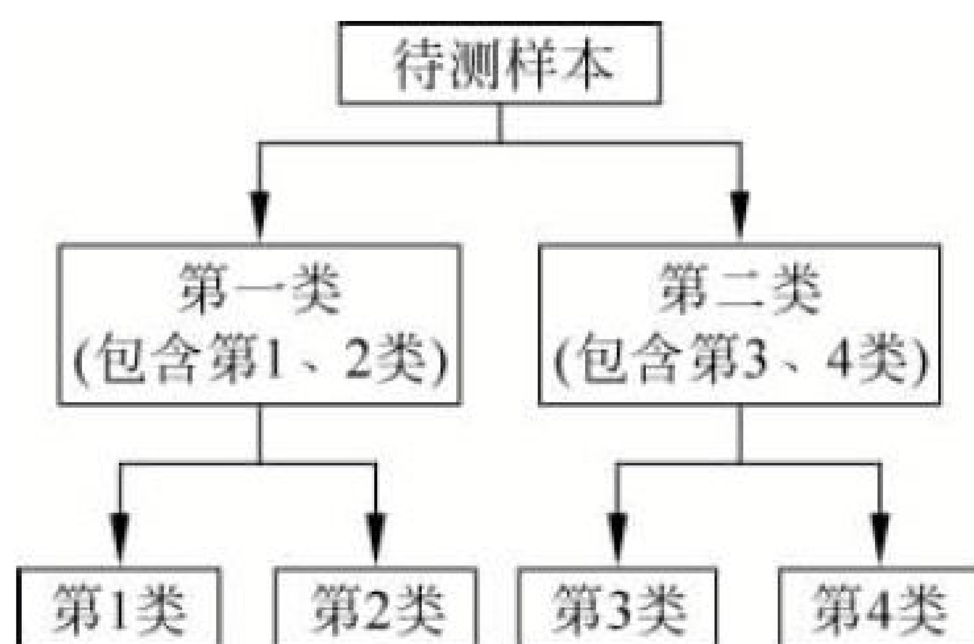


图 3-4 Fisher 分类流程图

将样本分成两大类有 3 种分法,如表 3-2 所示。

表 3-2 Fisher 分类方法

种 类	分 类 方 法
第一种	第 1、2 类作为第一类,第 3、4 类作为第二类
第二种	第 1、3 类作为第一类,第 2、4 类作为第二类
第三种	第 1、4 类作为第一类,第 2、3 类作为第二类

根据所给的训练样本数据,利用 MATLAB 程序得出训练样本分布图,如图 3-5 所示。

观察训练样本分布图可知,如果将第 1、2 类分在一起作为第一类,第 3、4 类分在一起作为第二类,这样很难将它们分开,因此排除这种分类方法,应选择第二、三种分类方法。

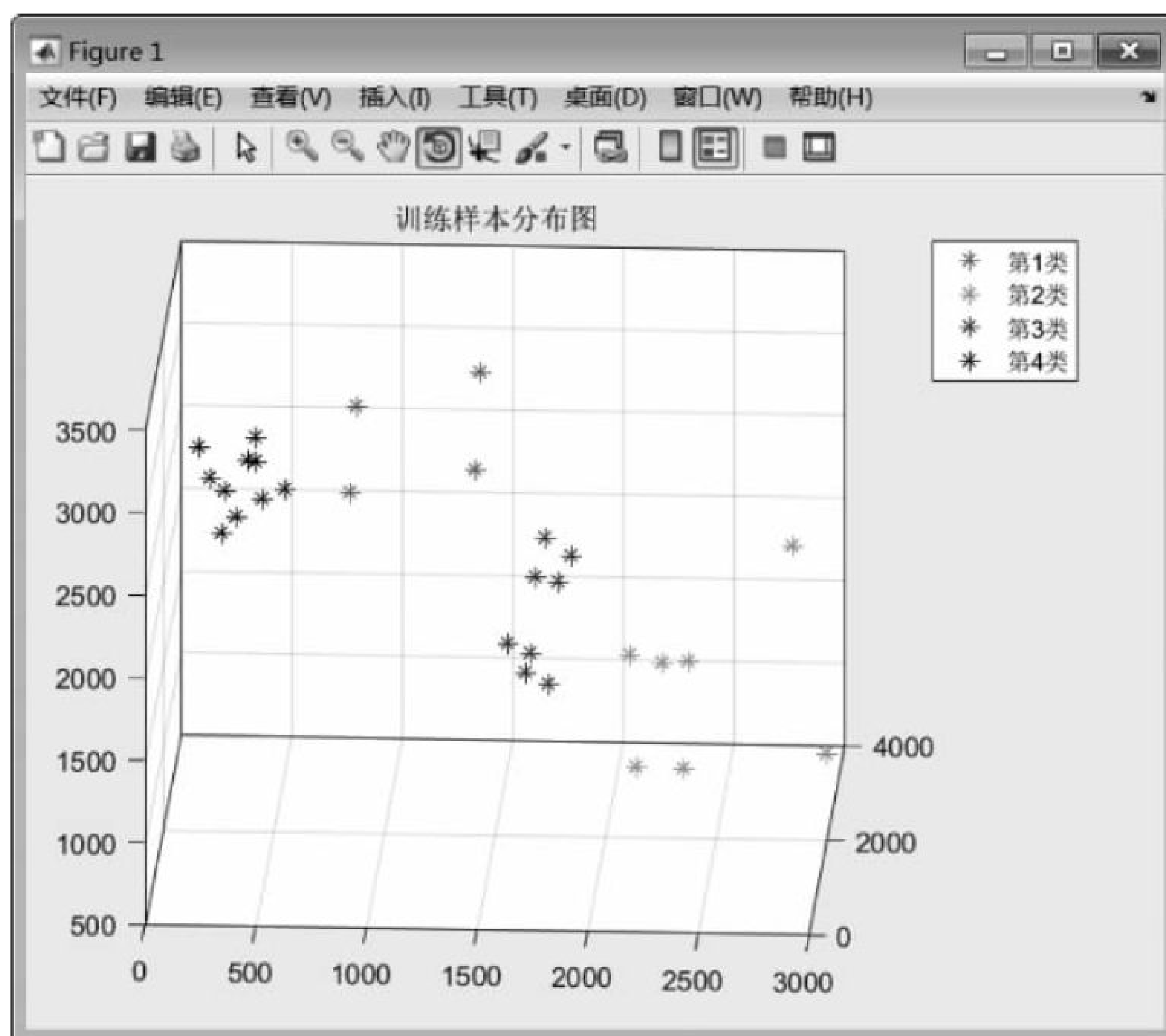


图 3-5 训练样本分布图

2. MATLAB 程序

(1) 选择第二种方法的相关程序及仿真结果。

训练样本分布图程序代码如下：

```
clear,close all;
N = 29;
X = [864.45     1647.31    2665.9;
     877.88     2031.66    3071.18;
     1418.79    1775.89    2772.9;
     1449.58    1641.58    3405.12;
     2352.12    2557.04    1411.53;
     2297.28    3340.14    535.62;
     2092.62    3177.21    584.32;
     2205.36    3243.74    1202.69;
     2949.16    3244.44    662.42;
     2802.88    3017.11    1984.98;
     2063.54    3199.76    1257.21;
     1739.94    1675.15    2395.96;
     1756.77    1652       1514.98;
     1803.58    1583.12    2163.05;
     1571.17    1731.04    1735.33;
     1845.59    1918.81    2226.49;
     1692.62    1867.5     2108.97;
     1680.67    1575.78    1725.1;
```

```
        1651.52     1713.28     1570.38;
        373.3       3087.05     2429.47;
        222.85      3059.54     2002.33;
        401.3       3259.94     2150.98;
        363.34      3477.95     2462.86;
        104.8       3389.83     2421.83;
        499.85      3305.75     2196.22;
        172.78      3084.49     2328.65;
        341.59      3076.62     2438.63;
        291.02      3095.68     2088.95;
        237.63      3077.78     2251.96;
        ]
fig = figure;
plot3(X(1:4,1),X(1:4,2), X(1:4,3),'r*')
hold on,plot3(X(5:11,1),X(5:11,2), X(5:11,3),'g*')
hold on,plot3(X(12:19,1),X(12:19,2), X(12:19,3),'b*')
hold on,plot3(X(20:29,1),X(20:29,2), X(20:29,3),'k*');grid;box
title('训练样本分布图')
legend('第 1 类','第 2 类','第 3 类','第 4 类')
```

测试样本分为一(1、3)类、二(2、4)类程序代码如下：

```
clear,close all;
N = 29;
X = [864.45     1647.31     2665.9
     877.88     2031.66     3071.18
     1418.79    1775.89     2772.9
     1449.58    1641.58     3405.12
     1739.94    1675.15     2395.96
     1756.77    1652        1514.98
     1803.58    1583.12     2163.05
     1571.17    1731.04     1735.33
     1845.59    1918.81     2226.49
     1692.62    1867.5      2108.97
     1680.67    1575.78     1725.1
     1651.52    1713.28     1570.38
     2352.12    2557.04     1411.53
     2297.28    3340.14     535.62
     2092.62    3177.21     584.32
     2205.36    3243.74     1202.69
     2949.16    3244.44     662.42
     2802.88    3017.11     1984.98
     2063.54    3199.76     1257.21
     373.3      3087.05     2429.47
     222.85     3059.54     2002.33
     401.3      3259.94     2150.98
     363.34     3477.95     2462.86
     104.8      3389.83     2421.83
     499.85     3305.75     2196.22
     172.78     3084.49     2328.65
     341.59     3076.62     2438.63
     291.02     3095.68     2088.95
     237.63     3077.78     2251.96]
```

```
fig = figure;
plot3(X(1:12,1),X(1:12,2), X(1:12,3),'b+')
hold on,plot3(X(13:29,1),X(13:29,2), X(13:29,3),'r+');grid;box
title('分为一、二类分布图')
m1 = mean(X(1:12,:));
m2 = mean(X(13:29,:));
S1 = 0;S2 = 0;
for i = 1:12
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 13:29
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [1702.8   1639.79   2068.74
    1877.93   1860.96   1975.3
    867.81    2334.68   2535.1
    1831.49   1713.11   1604.68
    460.69    3274.77   2172.99
    2374.98   3346.98   975.31
    2271.89   3482.97   946.7
    1783.64   1597.99   2261.31
    198.83    3250.45   2445.08
    1494.63   2072.59   2550.51
    1597.03   1921.52   2126.76
    1598.93   1921.08   1623.33
    1243.13   1814.07   3441.07
    2336.31   2640.26   1599.63
    354       3300.12   2373.61
    2144.47   2501.62   591.51
    426.31    3105.29   2057.8
    1507.13   1556.89   1954.51
    343.07    3271.72   2036.94
    2201.94   3196.22   935.53
    2232.43   3077.87   1298.87
    1580.1    1752.07   2463.04
    1962.4    1594.97   1835.95
    1495.18   1957.44   3498.02
    1125.17   1594.39   2937.73
    24.22     3447.31   2145.01
    1269.07   1910.72   2701.97
    1802.07   1725.81   1966.35
    1817.36   1927.4    2328.79
    1860.45   1782.88   1875.13];
y0 = W * (m1 + m2)'/2;
for i = 1:30
if W * x(i,:)'> y0
  disp('一')
```

```
  hold on,plot3(x(i,1),x(i,2),x(i,3),'g * ','MarkerSize',6,'LineWidth',2)
else
  disp('二')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'k * ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本一类','训练样本二类','测试样本一类','测试样本二类')
```

程序运行完之后,出现如图 3-6 所示的一、二类数据分类结果图界面。

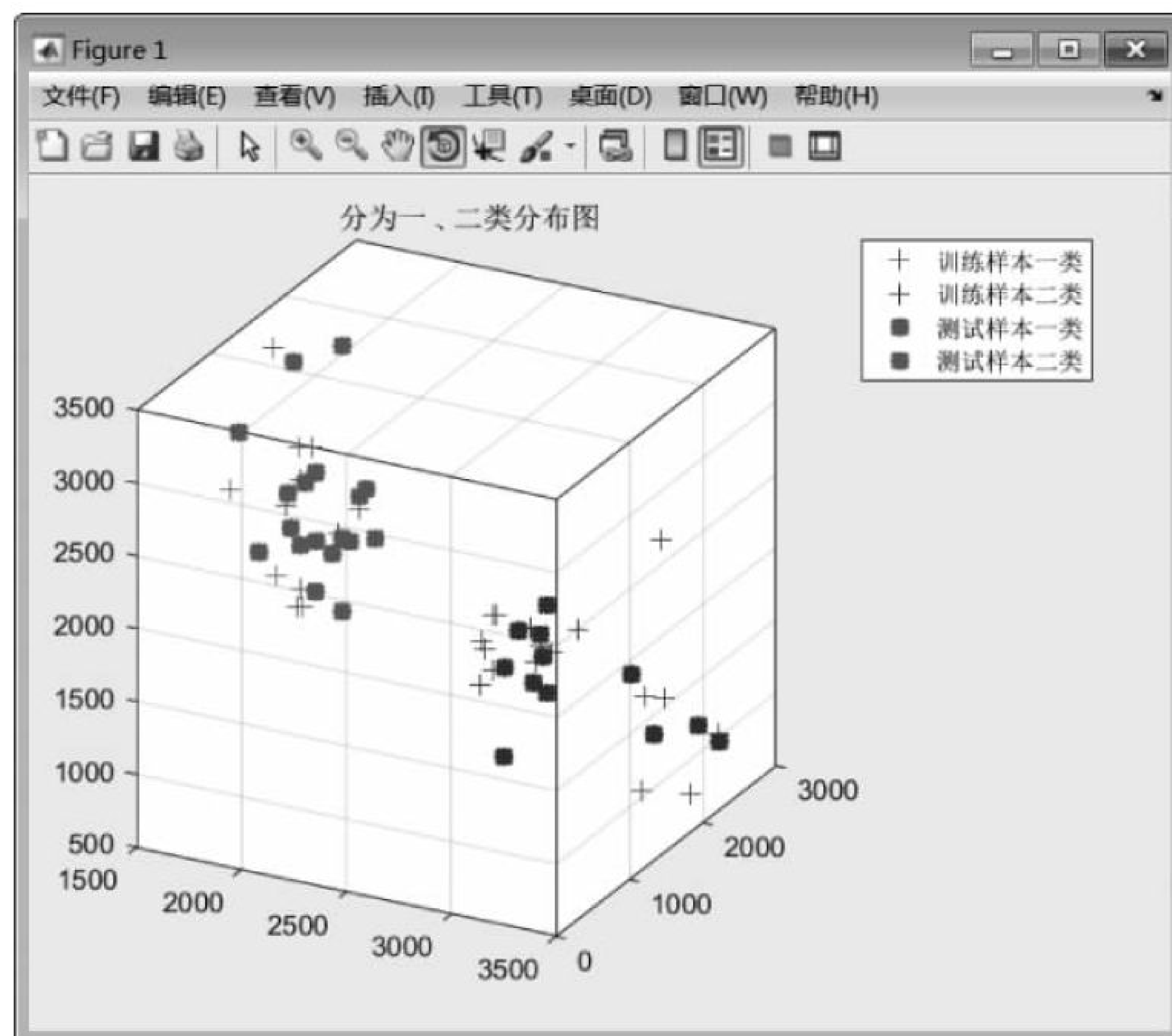


图 3-6 数据的一、二类分类结果图界面

运行 MATLAB 程序的结果如下:

```
W =
    0.2363   -0.9187   0.3166
一
一
一
一
二
二
二
一
二
一
一
一
```

```
一
二
二
二
二
一
二
二
二
一
一
一
一
二
一
一
一
一
```

测试样本一类分为1、3类的程序代码如下：

```
N = 12;
X = [864.45    1647.31    2665.9
     877.88    2031.66    3071.18
     1418.79   1775.89    2772.9
     1449.58   1641.58    3405.12
     1739.94   1675.15    2395.96
     1756.77   1652       1514.98
     1803.58   1583.12    2163.05
     1571.17   1731.04    1735.33
     1845.59   1918.81    2226.49
     1692.62   1867.5     2108.97
     1680.67   1575.78    1725.1
     1651.52   1713.28    1570.38]
fig = figure;
plot3(X(1:4,1),X(1:4,2), X(1:4,3),'r.')
hold on,plot3(X(5:12,1),X(5:12,2), X(5:12,3),'b.');grid;box
title('分为1、3类分布图')
m1 = mean(X(1:4,:));
m2 = mean(X(5:12,:));
S1 = 0;S2 = 0;
for i = 1:4
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 5:12
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
```

```
W = W./norm(W)
x = [1702.8    1639.79    2068.74
    1877.93    1860.96    1975.3
    867.81     2334.68    2535.1
    1831.49    1713.11    1604.68
    1783.64    1597.99    2261.31
    1494.63    2072.59    2550.51
    1597.03    1921.52    2126.76
    1598.93    1921.08    1623.33
    1243.13    1814.07    3441.07
    1507.13    1556.89    1954.51
    1580.1     1752.07    2463.04
    1962.4     1594.97    1835.95
    1495.18    1957.44    3498.02
    1125.17    1594.39    2937.73
    1269.07    1910.72    2701.97
    1802.07    1725.81    1966.35
    1817.36    1927.4     2328.79
    1860.45    1782.88    1875.13
];
  hold on,plot3(1495.18,1957.44,3498.02,'r+','MarkerSize',6,'LineWidth',2)
  hold on,plot3(1557.27,1746.27,1879.13,'b+','MarkerSize',6,'LineWidth',2)
y0 = W*(m1 + m2)'/2;
for i = 1:18
if W*x(i,:)'> y0
  disp('1')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r+','MarkerSize',6,'LineWidth',2)
else
  disp('3')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b+','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 1 类','训练样本 3 类','测试样本 1 类','测试样本 3 类')
```

程序运行完之后，出现如图 3-7 所示的 1、3 类数据分类结果图界面。

运行 MATLAB 程序的结果如下：

```
W =
    -0.4737    0.0499    0.8793
3
3
1
3
3
1
3
3
1
```

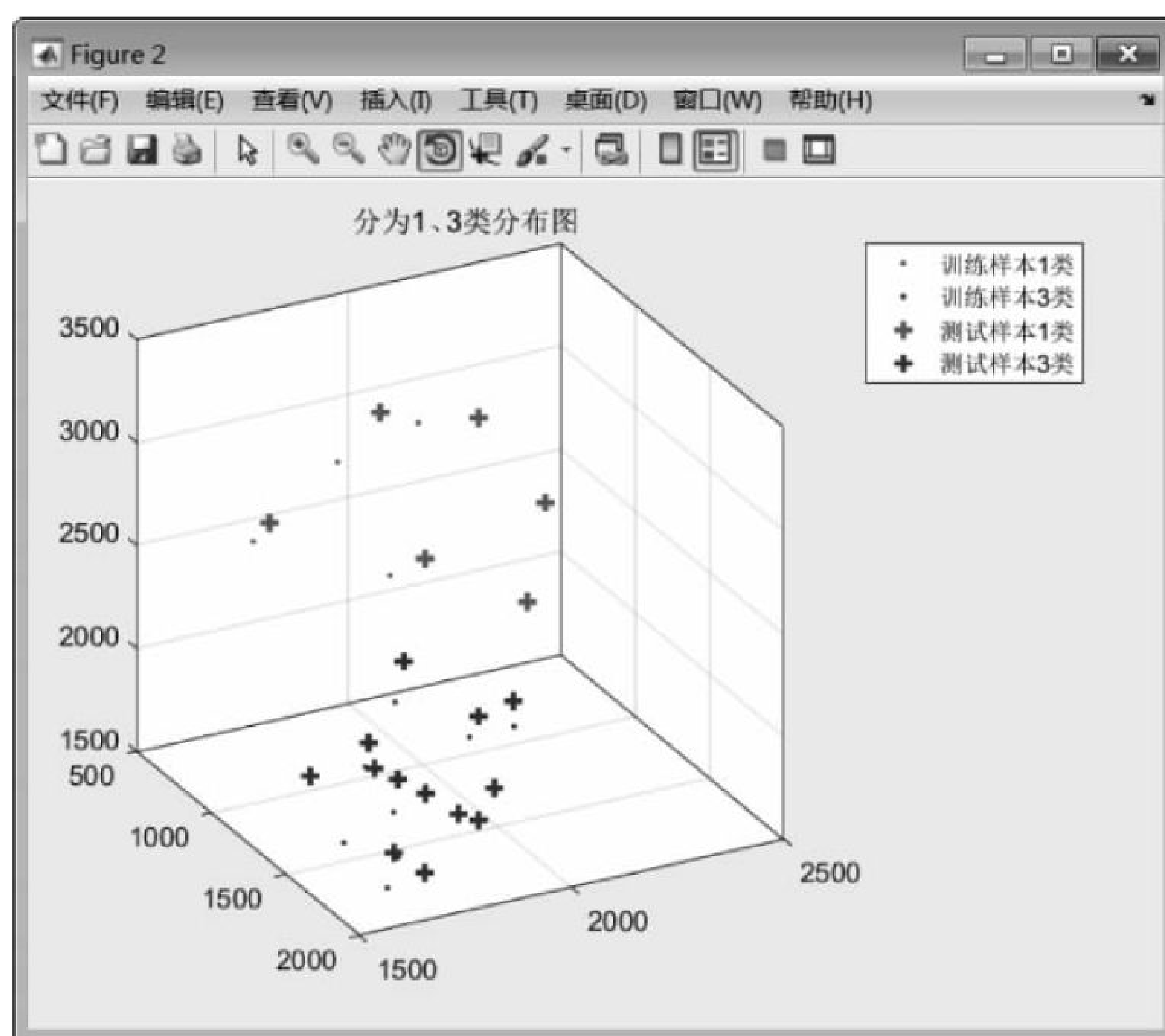


图 3-7　测试样本的 1、3 类分类结果图界面

```
3
3
3
1
1
1
3
3
3
```

测试样本二类分为 2、4 类的程序代码如下：

```
clear,close all;
N = 17;
X = [2352.12    2557.04    1411.53;
     2297.28    3340.14    535.62;
     2092.62    3177.21    584.32;
     2205.36    3243.74    1202.69;
     2949.16    3244.44    662.42;
     2802.88    3017.11    1984.98;
     2063.54    3199.76    1257.21;
     373.3      3087.05    2429.47;
     222.85     3059.54    2002.33;
     401.3      3259.94    2150.98;
     363.34     3477.95    2462.86;
```

```
        104.8       3389.83     2421.83;
        499.85      3305.75     2196.22;
        172.78      3084.49     2328.65;
        341.59      3076.62     2438.63;
        291.02      3095.68     2088.95;
        237.63      3077.78     2251.96;]
fig = figure;
plot3(X(1:7,1),X(1:7,2), X(1:7,3),'r.')
hold on,plot3(X(8:17,1),X(8:17,2), X(8:17,3),'b.');grid;box
title('分为 2、4 类分布图')
m1 = mean(X(1:7,:));
m2 = mean(X(8:17,:));
S1 = 0;S2 = 0;
for i = 1:7
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 8:17
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [460.69      3274.77     2172.99
    2374.98     3346.98     975.31
    2271.89     3482.97     946.7
    198.83      3250.45     2445.08
    2336.31     2640.26     1599.63
    354         3300.12     2373.61
    2144.47     2501.62     591.51
    426.31      3105.29     2057.8
    343.07      3271.72     2036.94
    2201.94     3196.22     935.53
    2232.43     3077.87     1298.87
    24.22       3447.31     2145.01
];
  hold on,plot3(2232.43,3077.87,1298.87,'r + ','MarkerSize',6,'LineWidth',2)
  hold on,plot3(362.51,3150.03,2472,'b + ','MarkerSize',6,'LineWidth',2)
y0 = W * (m1 + m2)'/2;
for i = 1:12
if W * x(i,:)'> y0
  disp('2')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r + ','MarkerSize',6,'LineWidth',2)
else
  disp('4')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b + ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 2 类','训练样本 4 类','测试样本 2 类','测试样本 4 类')
```

程序运行完之后，出现如图 3-8 所示的 2、4 类数据分类结果图界面。

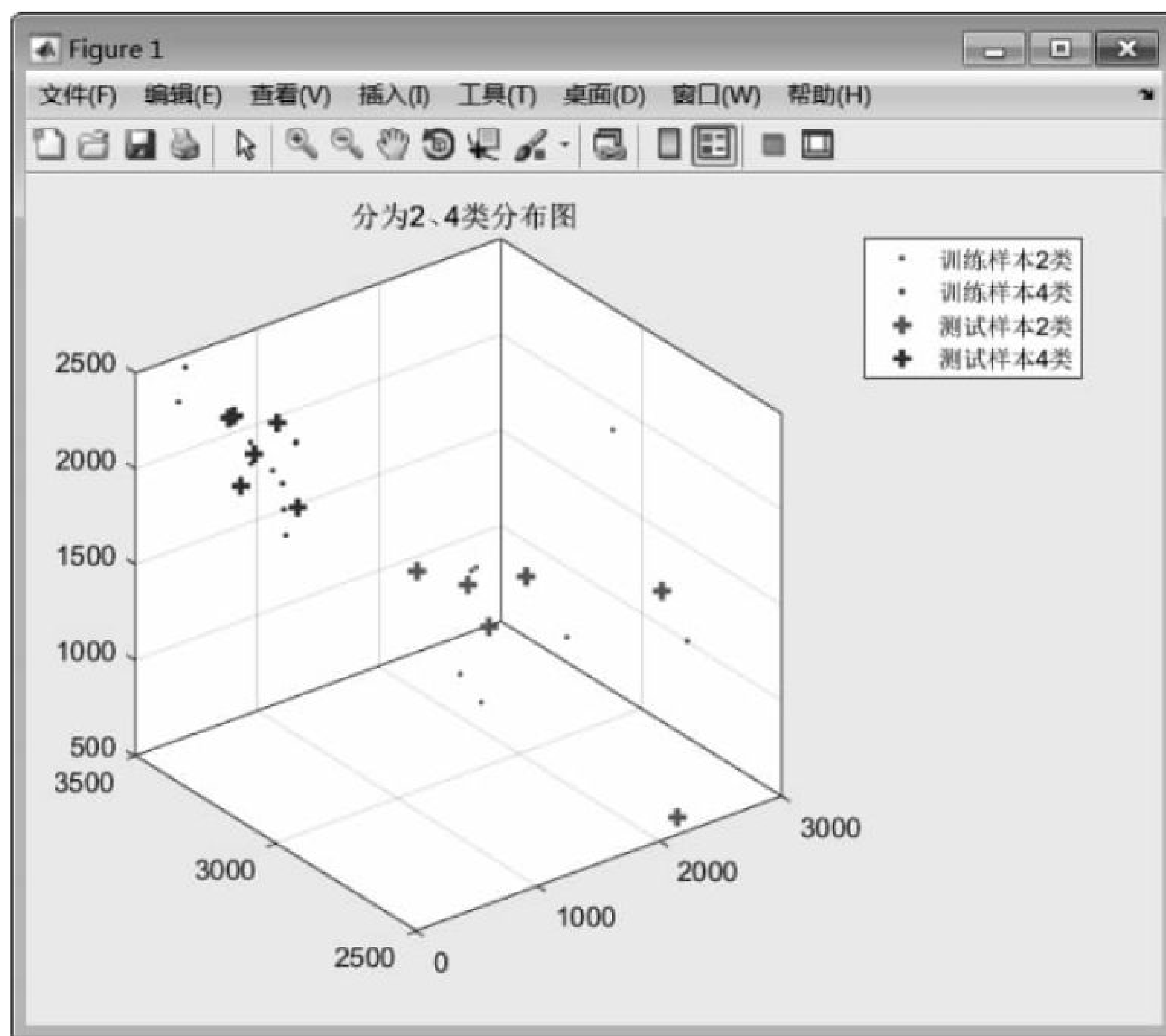


图 3-8 测试样本的 2、4 类分类结果图界面

运行 MATLAB 程序的结果如下：

```
W =
    0.8696   -0.0333   -0.4926
4
2
2
4
2
4
2
4
4
2
2
4
```

(2) 选择第三种方法的相关程序及仿真结果。

测试样本分为一(1、4)类、二(2、3)类的程序代码如下：

```
clear,close all;
N = 29;
X = [864.45    1647.31    2665.9
     877.88    2031.66    3071.18
     1418.79   1775.89    2772.9
     1449.58   1641.58    3405.12
     373.3     3087.05    2429.47;
     222.85    3059.54    2002.33;
```

```
        401.3      3259.94    2150.98;
        363.34     3477.95    2462.86;
        104.8      3389.83    2421.83;
        499.85     3305.75    2196.22;
        172.78     3084.49    2328.65;
        341.59     3076.62    2438.63;
        291.02     3095.68    2088.95;
        237.63     3077.78    2251.96;
        2352.12    2557.04    1411.53;
        2297.28    3340.14    535.62;
        2092.62    3177.21    584.32;
        2205.36    3243.74    1202.69;
        2949.16    3244.44    662.42;
        2802.88    3017.11    1984.98;
        2063.54    3199.76    1257.21;
        1739.94    1675.15    2395.96;
        1756.77    1652       1514.98;
        1803.58    1583.12    2163.05;
        1571.17    1731.04    1735.33;
        1845.59    1918.81    2226.49;
        1692.62    1867.5     2108.97;
        1680.67    1575.78    1725.1;
        1651.52    1713.28    1570.38;
        ]
fig = figure;
plot3(X(1:14,1),X(1:14,2), X(1:14,3),'r + ')
hold on,plot3(X(15:29,1),X(15:29,2), X(15:29,3),'b + ');grid;box
title('分为一、二类分布图')
m1 = mean(X(1:14,:));
m2 = mean(X(15:29,:));
S1 = 0;S2 = 0;
for i = 1:14
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 15:29
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [1702.8     1639.79    2068.74
    1877.93    1860.96    1975.3
    867.81     2334.68    2535.1
    1831.49    1713.11    1604.68
    460.69     3274.77    2172.99
    2374.98    3346.98    975.31
    2271.89    3482.97    946.7
    1783.64    1597.99    2261.31
    198.83     3250.45    2445.08
    1494.63    2072.59    2550.51
    1597.03    1921.52    2126.76
    1598.93    1921.08    1623.33
    1243.13    1814.07    3441.07
    2336.31    2640.26    1599.63
    354        3300.12    2373.61
    426.31     3105.29    2057.8
```

```
        2144.47     2501.62     591.51
        1507.13     1556.89     1954.51
        343.07      3271.72     2036.94
        2201.94     3196.22     935.53
        2232.43     3077.87     1298.87
        1580.1      1752.07     2463.04
        1962.4      1594.97     1835.95
        1495.18     1957.44     3498.02
        1125.17     1594.39     2937.73
        24.22       3447.31     2145.01
        1269.07     1910.72     2701.97
        1802.07     1725.81     1966.35
        1817.36     1927.4      2328.79
        1860.45     1782.88     1875.13];
y0 = W * (m1 + m2)'/2;
for i = 1:30
if W * x(i,:)'> y0
  disp('一')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r * ','MarkerSize',6,'LineWidth',2)
else
  disp('二')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b * ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本一类','训练样本二类','测试样本一类','测试样本二类')
```

程序运行完之后，出现如图 3-9 所示的数据分类结果图界面。

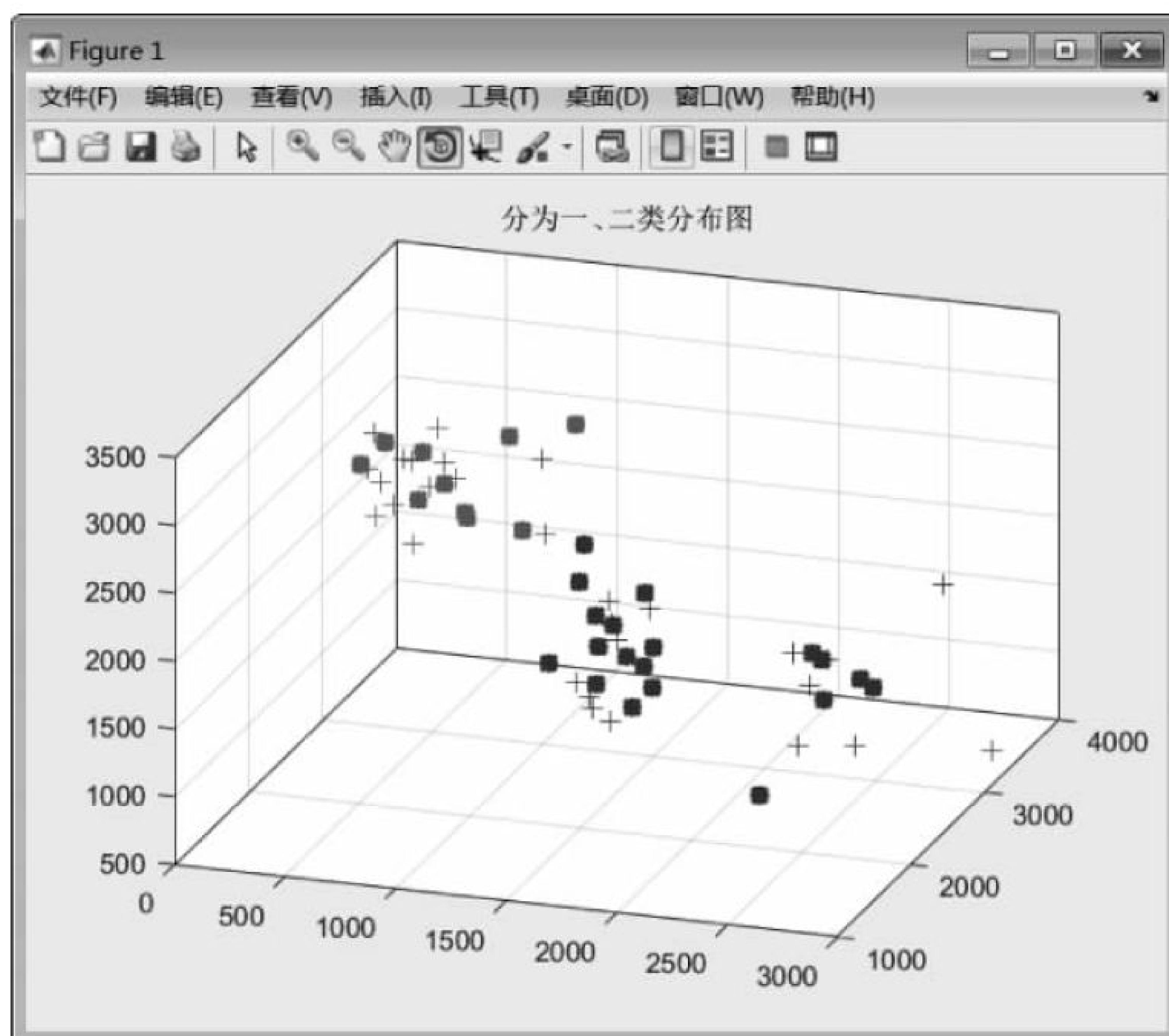


图 3-9　按第三种方法分为一、二类的结果图界面

运行 MATLAB 程序之后的结果如下：

```
W =
    -0.8228    0.2321    0.5188
二
二
一
二
一
二
二
二
一
二
二
二
一
二
一
二
一
二
一
二
二
二
二
一
一
一
一
二
二
二
```

测试样本一类分为 1、4 类的程序代码如下：

```
N=14;
X = [864.45    1647.31    2665.9
     877.88    2031.66    3071.18
     1418.79   1775.89    2772.9
     1449.58   1641.58    3405.12
     373.3     3087.05    2429.47;
     222.85    3059.54    2002.33;
     401.3     3259.94    2150.98;
     363.34    3477.95    2462.86;
     104.8     3389.83    2421.83;
     499.85    3305.75    2196.22;
     172.78    3084.49    2328.65;
     341.59    3076.62    2438.63;
     291.02    3095.68    2088.95;
     237.63    3077.78    2251.96;
     ]
```

```
fig = figure;
plot3(X(1:4,1),X(1:4,2), X(1:4,3),'r.')
hold on,plot3(X(5:14,1),X(5:14,2), X(5:14,3),'b.');grid;box
title('分为 1、4 类分布图')
m1 = mean(X(1:4,:));
m2 = mean(X(5:14,:));
S1 = 0;S2 = 0;
for i = 1:4
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 5:14
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [867.81   2334.68  2535.1
    460.69   3274.77  2172.99
    198.83   3250.45  2445.08
    1243.13  1814.07  3441.07
    354      3300.12  2373.61
    426.31   3105.29  2057.8
    343.07   3271.72  2036.94
    1495.18  1957.44  3498.02
    1125.17  1594.39  2937.73
    24.22    3447.31  2145.01
    1269.07  1910.72  2701.97
];
  hold on,plot3(1495.18,1957.44,3498.02,'r + ','MarkerSize',6,'LineWidth',2)
  hold on,plot3(1557.27,1746.27,1879.13,'b + ','MarkerSize',6,'LineWidth',2)
y0 = W * (m1 + m2)'/2;
for i = 1:11
if W * x(i,:)'> y0
  disp('1')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r + ','MarkerSize',6,'LineWidth',2)
else
  disp('4')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b + ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 1 类','训练样本 4 类','测试样本 1 类','测试样本 4 类')
```

程序运行完之后，出现如图 3-10 所示的 1、4 类数据分类结果界面。

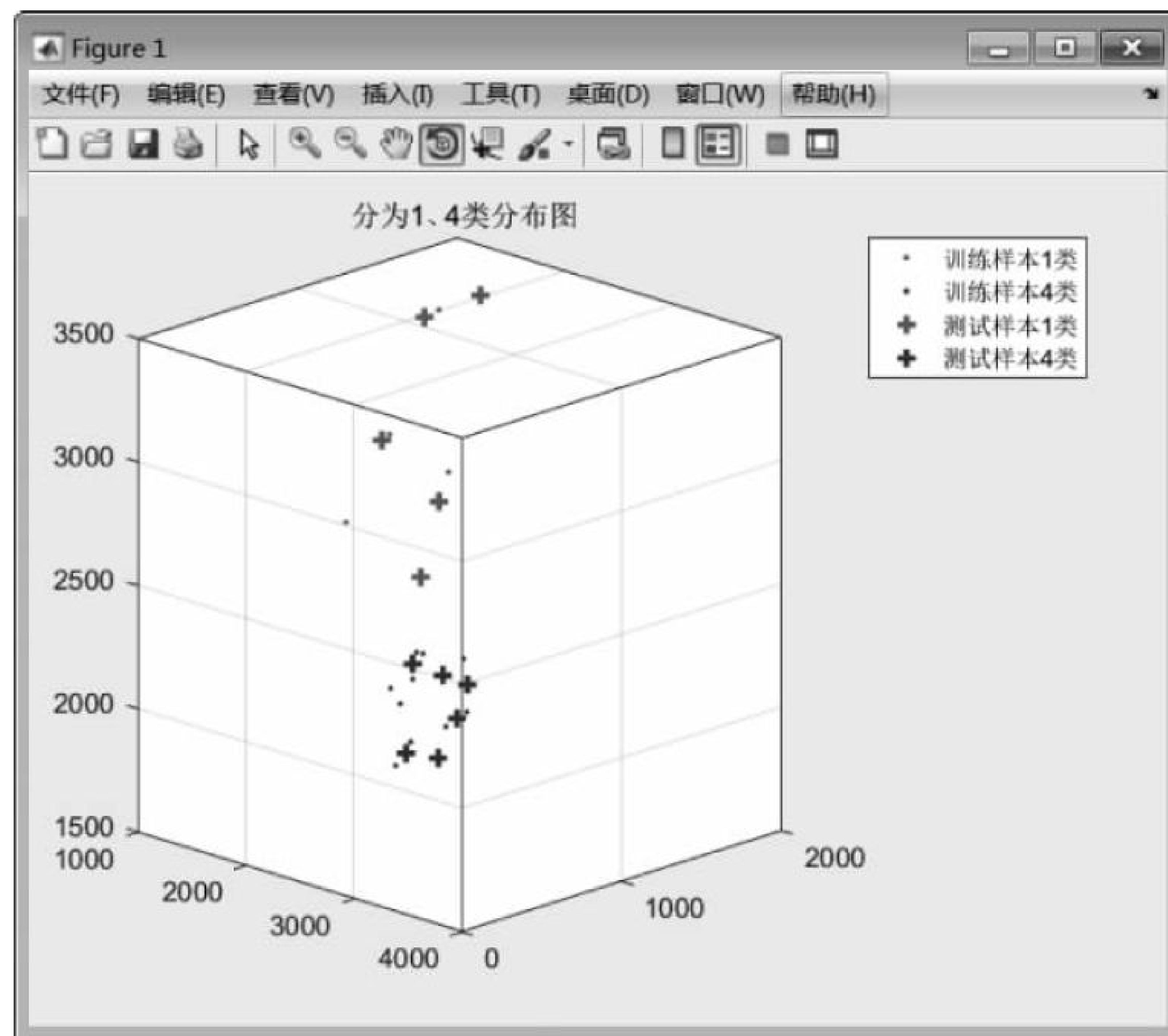


图 3-10 数据的 1、4 类分类结果图界面

运行 MATLAB 程序的结果如下：

```
W =
    0.4742   - 0.7890   0.3906
1
4
4
1
4
4
4
1
1
4
1
```

测试样本二类分为 2、3 类的程序代码如下：

```
clear,close all;
N = 15;
X = [2352.12    2557.04    1411.53;
     2297.28    3340.14    535.62;
     2092.62    3177.21    584.32;
     2205.36    3243.74    1202.69;
     2949.16    3244.44    662.42;
```

```
        2802.88     3017.11     1984.98;
        2063.54     3199.76     1257.21;
        1739.94     1675.15     2395.96
        1756.77     1652        1514.98
        1803.58     1583.12     2163.05
        1571.17     1731.04     1735.33
        1845.59     1918.81     2226.49
        1692.62     1867.5      2108.97
        1680.67     1575.78     1725.1
        1651.52     1713.28     1570.38
        ]
fig = figure;
plot3(X(1:7,1),X(1:7,2), X(1:7,3),'r.')
hold on,plot3(X(8:15,1),X(8:15,2), X(8:15,3),'b.');grid;box
title('分为2、3类分布图')
m1 = mean(X(1:7,:));
m2 = mean(X(8:15,:));
S1 = 0;S2 = 0;
for i = 1:7
    S1 = S1 + (X(i,:) - m1) * (X(i,:) - m1)';
end
for i = 8:15
    S2 = S2 + (X(i,:) - m2) * (X(i,:) - m2)';
end
Sw = S1 + S2;
W = inv(Sw) * (m1 - m2);
W = W./norm(W)
x = [1702.8         1639.79     2068.74
    1877.93     1860.96     1975.3
    1831.49     1713.11     1604.68
    2374.98     3346.98     975.31
    2271.89     3482.97     946.7
    1783.64     1597.99     2261.31
    1494.63     2072.59     2550.51
    1597.03     1921.52     2126.76
    1598.93     1921.08     1623.33
    2336.31     2640.26     1599.63
    2144.47     2501.62     591.51
    1507.13     1556.89     1954.51
    2201.94     3196.22     935.53
    2232.43     3077.87     1298.87
    1580.1      1752.07     2463.04
    1962.4      1594.97     1835.95
    1802.07     1725.81     1966.35
    1817.36     1927.4      2328.79
    1860.45     1782.88     1875.13
    ];
      hold on,plot3(2232.43,3077.87,1298.87,'r+','MarkerSize',6,'LineWidth',2)
      hold on,plot3(362.51,3150.03,2472,'b+','MarkerSize',6,'LineWidth',2)
    y0 = W * (m1 + m2)'/2;
```

```
for i = 1:19
if W * x(i,:)'> y0
  disp('2')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'r + ','MarkerSize',6,'LineWidth',2)
else
  disp('3')
  hold on,plot3(x(i,1),x(i,2),x(i,3),'b + ','MarkerSize',6,'LineWidth',2)
end
end
legend('训练样本 2 类','训练样本 3 类','测试样本 2 类','测试样本 3 类')
```

程序运行完之后,出现如图 3-11 所示的 2、3 类数据分类结果图界面。

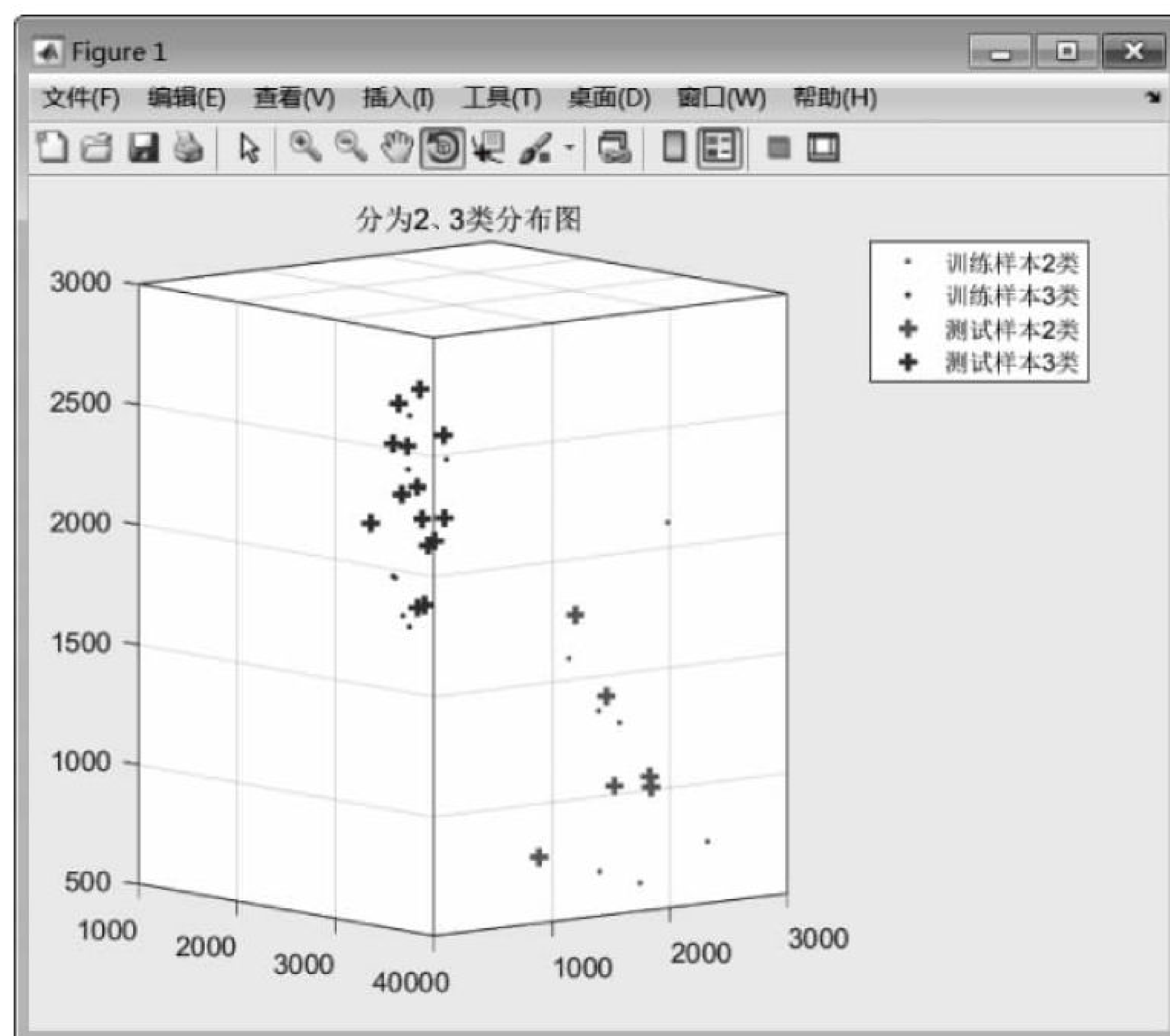


图 3-11 2、3 类数据分类结果图界面

运行 MATLAB 程序的结果如下:

```
W =
    0.3837    0.7917    - 0.4754
3
3
3
2
2
3
3
3
3
```

```
2
2
3
2
2
3
3
3
3
3
```

将两种分类方法的分类结果进行比较，即利用第二、三种分类方法进行比较，得出的分类结果如表 3-3 所示。

表 3-3 两种分类结果的比较

X	Y	Z	设计分类器分类结果 第二种(1、3,2、4 类)	设计分类器分类结果 第三种(1、4,2、3 类)
1702.8	1639.79	2068.74	3	3
1877.93	1860.96	1975.3	3	3
867.81	2334.68	2535.1	1	1
1831.49	1713.11	1604.68	3	3
460.69	3274.77	2172.99	4	4
2374.98	3346.98	975.31	2	2
2271.89	3482.97	946.7	2	2
1783.64	1597.99	2261.31	3	3
198.83	3250.45	2445.08	4	4
1494.63	2072.59	2550.51	1	3
1597.03	1921.52	2126.76	3	3
1598.93	1921.08	1623.33	3	3
1243.13	1814.07	3441.07	1	1
2336.31	2640.26	1599.63	2	2
354	3300.12	2373.61	4	4
2144.47	2501.62	591.51	2	2
426.31	3105.29	2057.8	1	4
1507.13	1556.89	1954.51	2	3
343.07	3271.72	2036.94	4	4
2201.94	3196.22	935.53	2	2
2232.43	3077.87	1298.87	3	2
1580.1	1752.07	2463.04	3	3
1962.4	1594.97	1835.95	3	3
1495.18	1957.44	3498.02	1	1
1125.17	1594.39	2937.73	1	1
24.22	3447.31	2145.01	1	4
1269.07	1910.72	2701.97	1	1
1802.07	1725.81	1966.35	3	3
1817.36	1927.4	2328.79	3	3
1860.45	1782.88	1875.13	3	3

比较这两种分类方法，方法二有两个错误分类，方法三仅有一个错误分类，所以方法三优于方法二。

3.5.6 结论

本节主要论述了 Fisher 分类法的概念、特点及其分类器设计，重点讨论了利用 Fisher 分类法设计分类器的全过程。在设计该种分类器的过程中，首先利用训练样本求得最佳投影方向 $\boldsymbol{w}^*$，并确定阈值点 y_0；接着通过分析来归纳给定样本数据的分类情况；最后利用 MATLAB 中的相关函数、工具设计了基于 Fisher 分类法的分类器，并对测试数据进行了成功分类。整个讨论和设计过程，关键点和创新点就在于对测试数据的处理过程上，通过两种方法做到了快速且相对准确的分类。

3.6 基于支持向量机的分类法

3.6.1 支持向量机简介

从观测数据中学习归纳出系统运动规律，并利用这些规律对未来数据或无法观测到的数据进行预测一直是智能系统研究的重点。传统学习方法中采用的经验风险最小化方法(ERM)虽然将误差最小化，但不能最小化学习过程的泛化误差。ERM 方法不成功的例子就是神经网络中的过学习问题。为此，由 Vapnik 领导的贝尔实验室研究小组于 1963 年提出了一种新的非常有潜力的分类技术，支持向量机(support vector machine，SVM)是一种基于统计学习理论的模式识别方法，主要应用于模式识别领域。

支持向量机的基本思想是在样本空间或特征空间构造出最优超平面，使得超平面与不同类样本集之间的距离最大，从而达到最大的泛化能力。

3.6.2 支持向量机基本思想

SVM 是从线性可分情况下的最优分类面方法发展而来的，基本思想可用图 3-12 的两类线性可分情况说明。在图 3-12 中，实心点和空心点代表两类样本，实线 P_0、P_1 为分类线。两条虚线分别为过各类中离分类线最近的样本且平行于分类线的直线，它们之间的距离叫作分类间隔。所谓最优分类线就是要求分类线不但能将两个类正确分开(训练错误率为0)，而且使分类间隔最大。

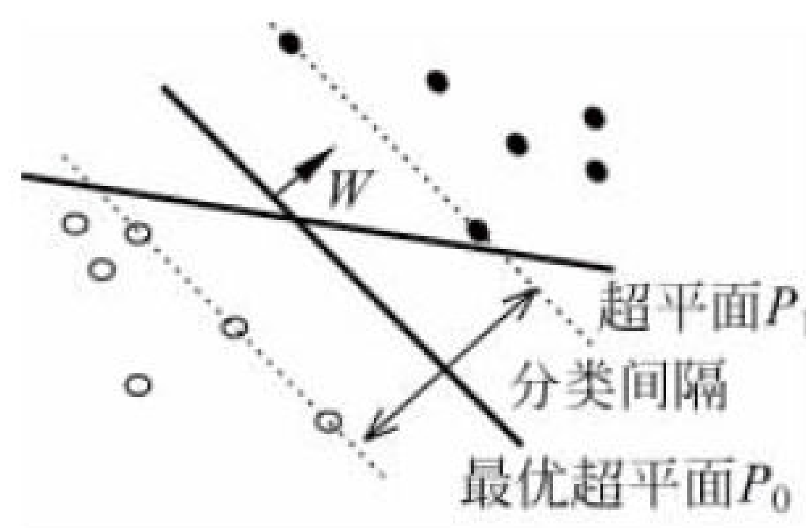


图 3-12 两类线性分割图

分类线方程为

$$\omega x + b = 0, \quad \omega \in R^m, b \in R \tag{3-33}$$

此时分类间隔为 $2/\|\omega\|$，使间隔最大等价于使 $\|\omega\|^2$ 最小，则可以通过求 $\|\omega\|^2/2$ 的极

小值获得分类间隔最大的最优超平面。

这里的约束条件为

$$y_i(\omega x_i + b) - 1 \geqslant 0, \quad i = 1,2,\cdots,n \tag{3-34}$$

该约束优化问题可以用 Lagrange 方法求解，令

$$L(\omega,b,\alpha) = \frac{1}{2}\|\omega\|^2 - \sum_{i=1}^{m}[y_i(\omega x_i + b) - 1] \tag{3-35}$$

其中 $\alpha_i \geqslant 0$ 为每个样本的拉氏乘子，由 L 分别对 b 和 ω 导数为 0，可以导出

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{3-36}$$

$$\omega = \sum_{i=1}^{m}\alpha_i y_i x_i \tag{3-37}$$

因此，解向量有一个由训练样本集的一个子集样本向量构成的展开式，该子集样本的拉氏乘子均不为 0，即支持向量。拉氏乘子为 0 的样本向量的贡献为 0，对选择分类超平面是无意义的。于是，就从训练集中得到了描述最优分类超平面的决策函数即支持向量机，它的分类功能由支持向量决定。这样决策函数可以表示为

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i (x + x_i) + b\right) \tag{3-38}$$

3.6.3 支持向量机的几个主要优点

(1) 它是专门针对有限样本情况的，其目标是得到现有信息下的最优解而不仅仅是样本数趋于无穷大时的最优值。

(2) 算法最终将转化成为一个二次型寻优问题。从理论上说，得到的将是全局最优点，解决了在神经网络方法中无法避免的局部极值问题。

(3) 算法将实际问题通过非线性变换转换到高维的特征空间(feature space)。在高维空间中构造线性判别函数来实现原空间中的非线性判别函数，特殊性质能保证机器有较好的推广能力，同时它巧妙地解决了维数问题，其算法复杂度与样本维数无关。

3.6.4 训练集为非线性情况

对于实际上难以线性分类的问题，待分类样本可以通过选择适当的非线性变换映射到某个高维的特征空间，使得在目标高维空间的这些样本线性可分，从而转化为线性可分问题。Cover 定理表明，通过这种非线性转换将非线性可分样本映射到足够高维的特征空间，非线性可分的样本将以极大的可能性变为线性可分。如果这个非线性转换为 $\phi(x)$，则超平面决策函数式可重写为

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i \phi(x)\phi(x_i) + b\right) \tag{3-39}$$

3.6.5 核函数

在上面的问题中只涉及训练样本之间的内积运算。实际上在高维空间只需进行内积运算,用原空间中的函数即可实现,甚至没有必要知道变换的形式。根据泛函的有关理论,只要一种 $K(x,x_i)$核函数满足 Mercer 条件,它就对应某一变换空间中的内积。因此,在最优分类面中采用适当的内积函数 $K(x,x_i)$就可以实现某一非线性变换后的线性分类,而计算复杂度却没有增加。核函数存在性定理表明:给定一个训练样本集,就一定存在一个相应的函数,训练样本通过核函数映射到高维特征空间的相是线性可分的。

对于一个特定的核函数,给定的样本集中的任意一个样本都可能成为一个支持向量。这意味着在一个支持向量机算法下观察到的特征在其他支持向量机算法下(其他核函数)并不能保持。因此对于解决具体问题来说,选择合适的核函数是很重要的。

常见的核函数有 3 类。

(1) 多项式核函数。

$$K(x,x_i) = [(x,x_i) + 1]^q \tag{3-40}$$

(2) 径向基函数(RBF)。

$$K(x,x_i) = \exp\left(\frac{|x - x_i|^2}{\sigma^2}\right) \tag{3-41}$$

(3) 采用 Sigmoid 函数作为内积。

$$K(x,x_i) = \tanh(v(x,x_i) + c) \tag{3-42}$$

3.6.6 多类分类问题

基本的支持向量机仅能解决两类分类问题。一些学者从两个方向研究用支持向量机解决多类分类问题:一个方向是将基本的两类支持向量机(Binary-class SVM,BSVM)扩展为多类分类支持向量机(multi-class SVM,MSVM),使支持向量机本身成为解决多类分类问题的多类分类器;另一方向则相反,将多类分类问题逐步转化为两类分类问题,即用多个两类分类支持向量机组成的多类分类器。

1. 多类分类支持向量机 MSVM

实际应用研究中多类分类问题更加常见,只要将目标函数由两类改为多类(k 类)情况,就可以很自然地将 BSVM 扩展为多类分类支持向量机 MSVM,以相似的方式可得到决策函数。

2. 基于 BSVM 的多类分类器

这种方案是为每个类构建一个 BSVM,如图 3-13 所示。对于每个类的 BSVM,其训练样本集的构成是:属于该类的样本为正样本,而不属于该类的其他所有样本为负样本,即该 BSVM 分类器将该类样本和其他样本分开。在 1-a-1 分类过程中训练样本需要重新标注,因为一个样本只有在对应类别的 BSVM 分类器才是正样本,对其他的 BSVM 分类器都是

负样本。

1) 1-a-1 分类器(One-against-one classifiers)

对于 1-a-1 分类器,解决 k 类分类问题就需要用到 BSVM,这种方案是每两个类别训练一个 BSVM 分类器,如图 3-14 所示。最后一个待识别样本的类别是由所有 $k(k-1)/2$ 个 BSVM"投票"决定的。

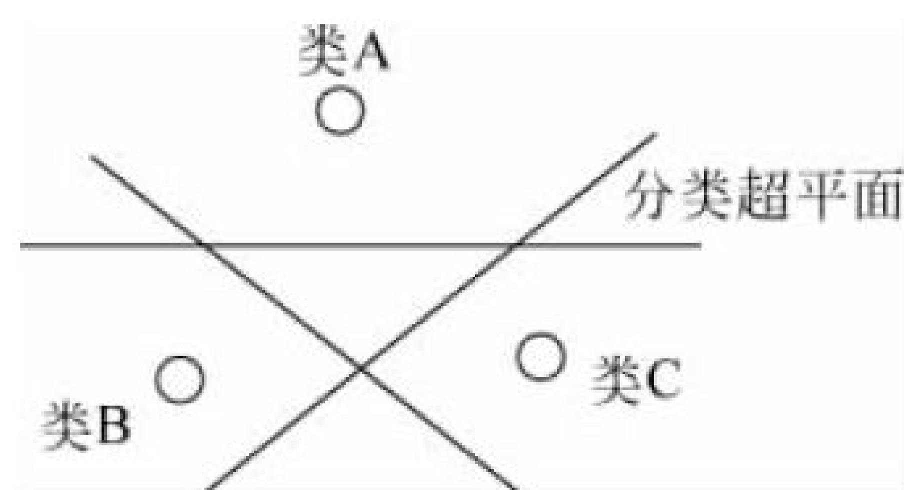


图 3-13　BSVM 分类原理图

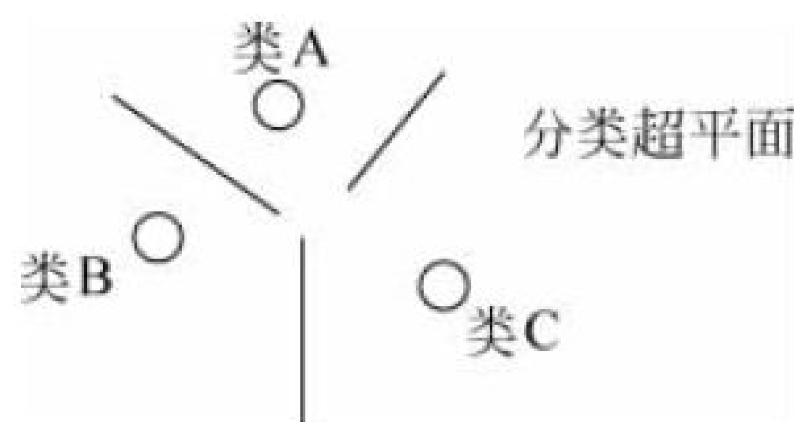


图 3-14　1-a-1 分类原理图

2) 多级 BSVM 分类器

这种方案是把多类分类问题分解为多级的两类分类子问题,如图 3-15 所示。两种典型方案中,A、B、C、D、E、F 分别表示 7 个不同的类。

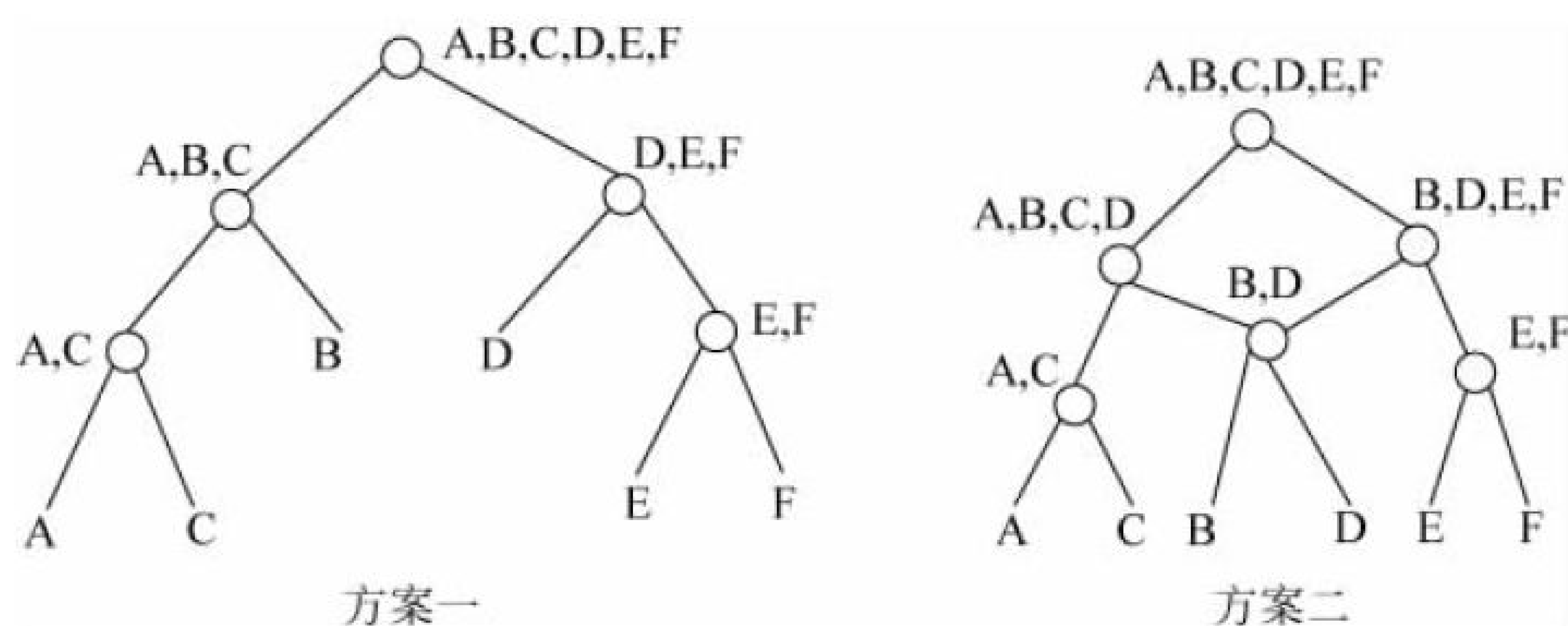


图 3-15　BSVM 多级分类

3.6.7　基于 SVM 的 MATLAB 实现

1. 建立模型流程图

基于 SVM 的数据分类设计流程如图 3-16 所示。



图 3-16　基于 SVM 的数据分类设计流程

2. 数据预处理

对训练集和测试集进行归一化预处理,采用[0,1]区间归一化。

$$f: x \rightarrow y = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

3. 训练和预测

以下是 MATLAB 中 LibSVM 工具箱中自带的 SVM 训练和预测的语句。

```
●model = svmtrain(train_labels, train_metrix,['libsvm_options'])
-- train_labels: 训练集的标签
-- train_metrix: 训练集的属性
-- libsvm_options: 一些选项参数
-- model: 训练得到的分类模型
●[predict_label, accuracy] = svmpredict(test_labels, test_metrix, model)
-- test_labels: 测试集的标签
-- test_metrix: 测试集的属性
-- model: 由 svmtrain 得到的分类模型
-- predict_label: 预测得到的测试集的标签
-- accuracy: 分类准确率
```

4. LibSVM 工具箱简介

LibSVM 是台湾大学林智仁教授等人开发设计的一个简单、易于使用且快速有效的 SVM 模式识别与回归的软件包。它不但提供了编译好的可在 Windows 系统中执行的文件，还提供了源代码，方便用户改进、修改以及在其他操作系统上应用。该软件还有一个特点，就是对 SVM 所涉及的参数调节相对比较少，提供了很多的默认参数，利用这些默认参数就可以解决很多问题；同时还提供了交互检验的功能。

以下是 LibSVM 工具箱的安装过程。

(1) 下载 libsvm-mat-2.91-1 的压缩文件，将其解压到 MATLAB→toolbox 的安装路径下。

(2) 在 MATLAB 软件中执行“设置路径”→“添加文件夹”命令，将该压缩包解压后添加到工具箱即可。

5. MATLAB 源程序

在程序开始时要将数据进行归一化处理，归一化程序如下：

```
function normal = normalization(x,kind)
%last modified 2009.2.24
%
if nargin < 2
    kind = 2; %kind = 1 or 2 表示第一类或第二类规范化
end

[m,n] = size(x);
normal = zeros(m,n);
%%normalize the data x to [0,1]
if kind == 1
    for i = 1:m
```

```
        ma = max( x(i,:) );
        mi = min( x(i,:) );
        normal(i,:) = ( x(i,:) - mi )./( ma - mi );
    end
end
%% normalize the data x to [ - 1,1]
if kind == 2
    for i = 1:m
        mea = mean( x(i,:) );
        va = var( x(i,:) );
        normal(i,:) = ( x(i,:) - mea )/va;
    end
end
```

SVM 的 MATLAB 完整源程序如下：

```
clear;
clc;
load SVM;

train_train = [train(1:4,:);train(5:11,:);train(12:19,:);train(20:30,:)]; % 手动划分为 4 类
train_target = [target(1:4);target(5:11);target(12:19);target(20:30)];
test_simulation = [simulation(1:6,:);simulation(7:11,:);simulation(12:24,:);
simulation(25:30,:)];
test_labels = [labels(1:6);labels(7:11);labels(12:24);labels(25:30)];

%train_train = normalization(train_train',2);
%test_simulation = normalization(test_simulation',2);
%train_train = train_train';
%test_simulation = test_simulation';

%           bestcv = 0;
%           for log2c = -10:10,
%           for log2g = -10:10,
%             cmd = ['-v 5 -c ', num2str(2^log2c), ' -g ',num2str(2^log2g)]; %将训练集分
                                                                          %为 5 类
%             cv = svmtrain(train_target, train_train, cmd);
%             if (cv >= bestcv),
%               bestcv = cv; bestc = 2^log2c; bestg = 2^log2g;
%             end
%           end
%         end
%         fprintf('(best c= %g, g= %g, rate= %g)\n',bestc, bestg, bestcv);
%         cmd = ['-c ', num2str(bestc), ' -g ', num2str(bestg)];
%         model = svmtrain(train_target, train_train, cmd);

model = svmtrain(train_target, train_train, '-c 2 -g 0.2 -t 1'); %核函数
  [predict_label, accuracy] = svmpredict(test_labels, test_simulation, model);
hold off
```

```
f = predict_label';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
plot3(simulation(:,1),simulation(:,2),simulation(:,3),'o');
line(simulation(index1,1),simulation(index1,2),simulation(index1,3),'linestyle','none',
'marker','*','color','g');
line(simulation(index2,1),simulation(index2,2),simulation(index2,3),'linestyle','none',
'marker','<','color','r');
line(simulation(index3,1),simulation(index3,2),simulation(index3,3),'linestyle','none',
'marker','+','color','b');
line(simulation(index4,1),simulation(index4,2),simulation(index4,3),'linestyle','none',
'marker','>','color','y');
box;grid on;hold on;
xlabel('A');
ylabel('B');
zlabel('C');
title('支持向量机分类图');
```

程序运行完之后，出现如图 3-17 所示的分类结果图界面。

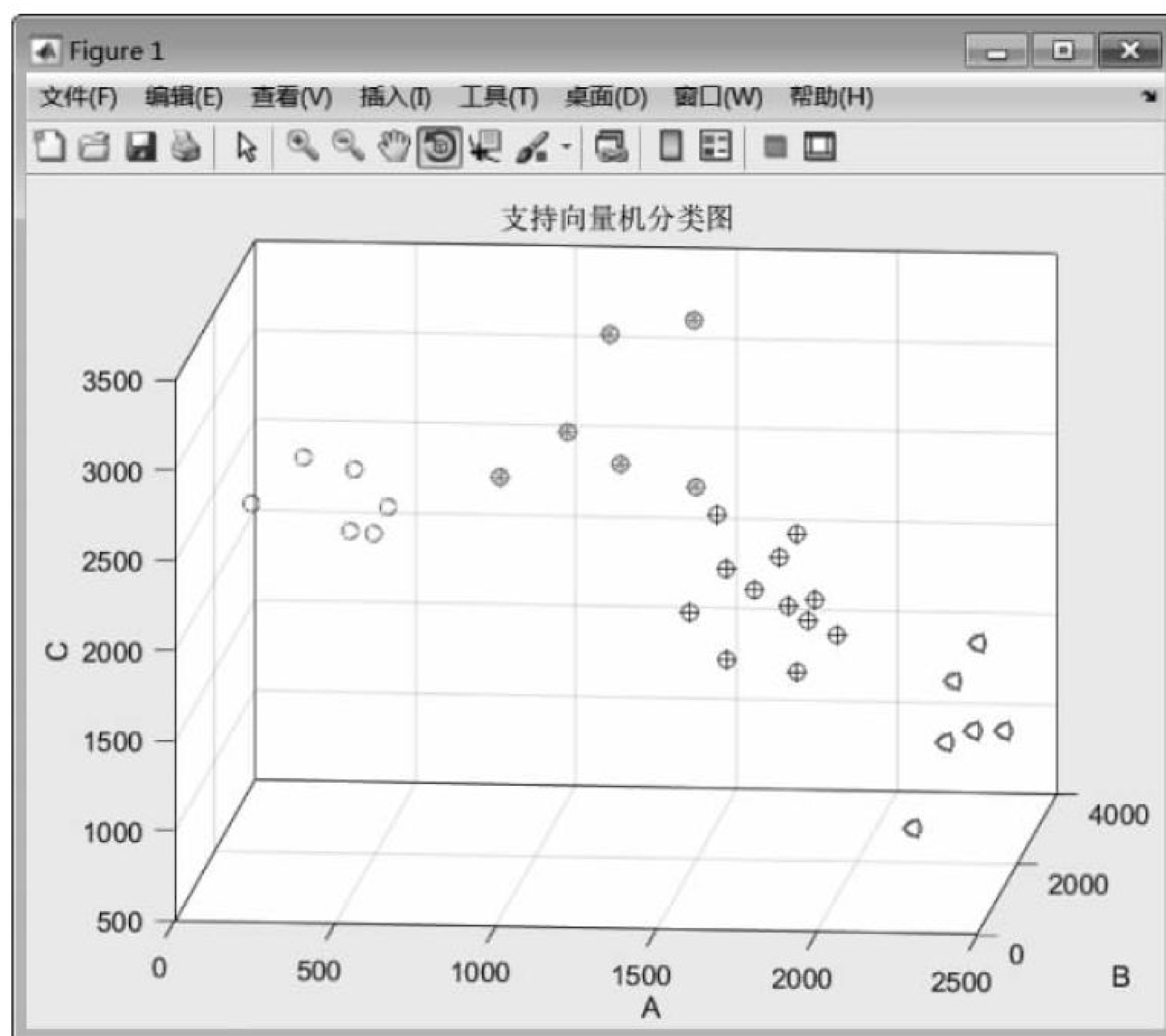


图 3-17 支持向量机分类结果图界面

在 MATLAB 命令窗口将出现如下结果：

```
Accuracy = 96.6667% (29/30) (classification)
predict_label =
```

```
        1
        1
        1
        1
        1
        1
        2
        2
        2
        2
        2
        3
        3
        3
        3
        3
        3
        2
        3
        3
        3
        3
        3
        3
        4
        4
        4
        4
        4
        4
```

从程序运行结果可以看出分类正确率为96.6667%,有一个样本分错了。

3.6.8 结论

SVM具有优良的学习能力和推广能力,能够有效克服"维数灾难"和"过学习"问题,而SVM的参数是影响分类精度、回归预测的重要因素。仿真结果表明,预测结果可靠,可以为数据提供很好的参考信息。

习题

(1) 什么是判别函数法?

(2) 怎样确定线性判别函数的系数?

(3) 线性判别函数的分类器设计方法有哪些?非线性判别函数的分类器设计方法有哪些?它们有什么异同?

第4章

聚类分析

4.1 聚类分析

聚类分析是指事先不了解一批样本中的每一个样本的类别或其他的先验知识，而唯一的分类根据是样本的特征，利用某种相似度度量的方法，把特征相同或相似的归为一类，实现聚类划分。

聚类分析就是对探测数据进行分类分析的一个工具，许多学科要根据所测得的或感知到的相似性对数据进行分类，把探测数据归入到各个聚合类中，且在同一个聚合类中的模式比不同聚合类中的模式更相似，从而对模式间的相互关系做出估计。聚类分析的结果可以被用来对数据提出初始假设，分类新数据，测试数据的同类型及压缩数据。

聚类算法的重点是寻找特征相似的聚合类。人是二维的最佳分类器，然而大多数实际的问题涉及高维的聚类，对高维空间内的数据的直观解释，其困难是显而易见的。另外，数据也不会服从规则现象分布，这就是有大量聚类算法出现在文献中的原因。

4.1.1 聚类的定义

Everett 提出，一个聚合类是一些相似的实体集合，而且不同聚合类的实体是不相似的。在一个聚合类内的两个点间的距离小于在这个类内任一点和不在这个类内的另一任一点间的距离。聚合类可以被描述成在 n 维空间内存在较高密度点的连续区域和较低密度点的区域，而较低密度点的区域把其他较高密度点的区域分开。

在模式空间 S 中，若给定 N 个样本 $X_1,X_2,\cdots,X_N$，聚类的定义是：按照相互类似的程度找到相应的区域 $R_1,R_2,\cdots,R_M$，对任意 $X_i(i=1,2,\cdots,N)$归入其中一类，而且不会同时属于两类，即

$$R_1 \cup R_2 \cup \cdots \cup R_M = R \tag{4-1}$$

$$R_i \cap R_j = \varnothing, \quad i \neq j \tag{4-2}$$

这里∩、∪分别指交集和并集。

选择聚类的方法应以一个理想的聚类概念为基础。然而,如果数据不满足由聚类技术所做的假设,则算法不是去发现真实的结构而是在数据上强加某种结构。

4.1.2 聚类准则

设有未知类别的 N 个样本,要把它们划分到 M 类中去,可以有多种优劣不同的聚类方法,怎样评价聚类的优劣,这就需要确定一种聚类准则。但客观地说,聚类的优劣是就某一种评价准则而言,很难有对各种准则均呈优良表现的聚类方法。

聚类准则的确定,基本上有两种方法。一种是试探法,根据所分类的问题,确定一种准则,并用它来判断样本分类是否合理。例如,以距离函数作为相似性的度量,用不断修改的阈值来探究对此种准则的满足程度,当取得极小值时,就认为得到了最佳划分。另一种是规定一种准则函数,其函数值与样本的划分有关,当取得极小值时,就认为得到了最佳划分下给出的一种简单而又广泛应用的准则,即误差平方和准则:

设有 N 个样本,分属于 $\omega_1,\omega_2,\cdots,\omega_M$ 类,设有 N_i 个样本的 ω_j 类,其均值为

$$m_i = \frac{1}{N_i}\sum_{i=1}^{N_i} X, \quad i = 1,2,\cdots,M \tag{4-3}$$

$$\overline{X^{(\omega_i)}} = \frac{1}{N_i}\sum_{i=1} X, \quad i = 1,2,\cdots,M \tag{4-4}$$

因为有若干种方法可将 N 个样本划分到 M 类中去,因此对应一种划分,可求得一个误差平方和 J,要找到使 J 值最小的那种划分。定义误差平方和

$$J = \sum_{i=1}^{M}\sum_{k=1}^{N_i} \| X_k - m_i \|^2 \tag{4-5}$$

$$J = \sum_{i=1}^{M}\sum_{k=1}^{N_i} \| X_k - \overline{X^{(\omega_i)}} \|^2 \tag{4-6}$$

经验表明,当各类样本均很密集,各类样本个数相差不大,而类间距离较大时,适合采用误差平方和准则。若各类样本数相差很大,类间距离较小时,就有可能将样本数多的类一分为二,而得到的 J 值却比大类保存完整时小,误以为得到了最优划分,实际上得到了错误分类。

4.1.3 基于试探法的聚类设计

基于试探法的聚类设计采用假设某种分类方案,确定一种聚类准则,计算 J 值,找到 J 值最小的那一种分类方案,则认为该种方法为最优分类。基于试探的未知类别聚类算法,包括最临近规则的试探法、最大最小距离试探法和层次聚类试探法。

1. 最临近规则的试探法

假设前 i 个样本已经被分到 k 个类中。则第 $i+1$ 个样本应该归入哪一个类?假设归入 ω_a 类,要使 J 最小,则应满足第 $i+1$ 个样本到 ω_a 类的距离小于给定的阈值;若大于给定

的阈值 T,则应为其建立一个新的类 ω_{k+1}。在未将所有的样本分类前,类数是不能确定的。

这种算法与第一个中心的选取、阈值 T 的大小、样本排列次序及样本分布的几何特性有关。这种方法运算简单,当用有关于模式几何分布的先验知识作指导给出阈值 T 及初始点时,则能较快地获得合理的聚类结果。

2. 最大最小距离试探法

最临近规则的试探法受阈值 T 的影响很大。阈值的选取是聚类成败的关键之一。最大最小距离算法充分利用样本内部特性,计算出所有样本间的最大距离作为归类阈值的参考,改善了分类的准确性。例如,采用某样本到某一个聚类中心的距离小于最大距离的一半,则归入该类,否则建立新的聚类中心。

3. 层次聚类试探法

层次聚类方法对给定的数据集进行层次的分解,直到某种条件满足为止。具体又可分为合并、分裂两种方案。

合并的层次聚类是一种自底向上的策略,首先将每个对象作为一个类,然后根据类间距离的不同,合并距离小于阈值的类,合并一些相似的样本,直到终结条件被满足。合并算法会在每一步减小聚类中心数量,聚类产生的结果来自于前一步的两个聚类的合并。绝大多数层次聚类方法属于这一类,它们只是在相似度的定义上有所不同。

分裂的层次聚类与合并的层次聚类相反,采用自顶向下的策略,它首先将所有对象置于同一个簇中,然后逐渐细分为越来越小的样本簇,直到达到了某个终止条件。分裂算法与合并算法的原理相反,在每一步增加聚类中心数目,每一步聚类产生的结果,都是将前一步的一个聚类中心分裂成两个而得到的。

常用的聚类方法有均值聚类、分层聚类和模糊聚类。

4.2 数据聚类——K 均值聚类

4.2.1 K 均值聚类简介

K 均值聚类发明于 1956 年,该算法最常见的形式是采用被称为劳埃得算法(Lloyd algorithm)的迭代式改进探索法。劳埃得算法首先把输入点分至 K 个初始化分组,可以使用随机或者一些启发式数据,然后计算每组的中心点,根据中心点的位置把样本分到离它最近的中心,重新确定分组,再继续重复不断地计算中心并重新分组,直到收敛,即样本不再改变分组(中心点位置不再改变)。

4.2.2 K 均值聚类原理

动态聚类方法是模式识别中一种普遍采用的方法,它具有以下 3 个要点:

(1) 选定某种距离度量作为样本间的相似性度量。

(2) 确定某个评价聚类结果质量的准则函数。

(3) 给定某个初始分类，然后用迭代算法找出使准则函数取极值的最好的聚类结果。

K 均值算法是基于质心的技术，K 均值算法以 K 为输入参数，把 n 个对象集合分为 K 个簇，使得簇内的相似度高，簇之间的相似度低。簇的相似度是关于簇中对象的均值度量，可以看作是簇的质心。

K 均值聚类算法使用的聚类准则函数是误差平方和准则 J_K

$$J_K = \sum_{j=1}^{K} \sum_{k=1}^{n_j} \| x_k - m_j \|^2 \tag{4-7}$$

为了使聚类结果优化，应该使准则 J_K 最小化。

(1) 给出 n 个混合样本，令 $I=1$，表示迭代次数，选取 K 个初始聚合中心 $Z_j(I)$，$j=1,2,\cdots,K$。

(2) 计算每个样本与聚合中心的距离 $D[x_k, Z_j(I)]$($k=1,2,\cdots,n$；$j=1,2,\cdots,K$)。

若 $D[x_k, Z_i(I)] = \min\limits_{j=1,2,\cdots,c} \{D[x_k, Z_j(I)], k=1,2,\cdots,n\}$，则 $x_k \in \omega_i$。

(3) 计算 K 个新的集合中心：$Z_j(I+1) = \dfrac{1}{n_j} \sum\limits_{k=1}^{n_j} x_k^{(j)}$ ($j=1,2,\cdots,K$)。

(4) 判断：若 $Z_j(I+1) \neq Z_j(I)$ ($j=1,2,\cdots,K$)，则 $I=I+1$，返回(2)；否则，算法结束。

K 均值算法的执行过程如图 4-1 所示。

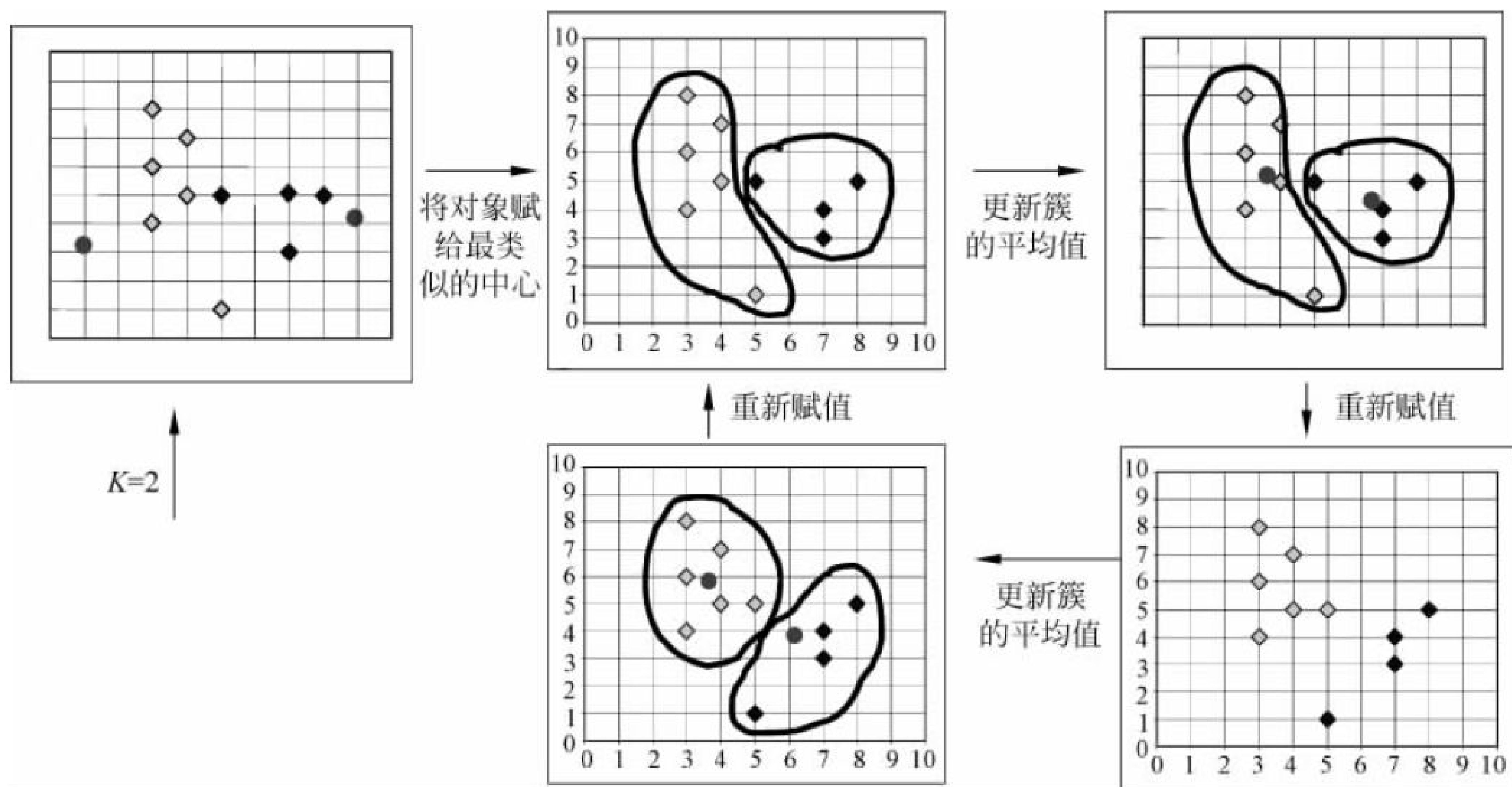


图 4-1 K 均值聚类算法执行过程

接下来介绍初始分类的选取和调整的方法。

(1) 代表点也就是聚类中心。选取一批代表点，计算其他样本到聚类中心的距离，把所有样本归于最近的聚类中心点，形成初始分类，再重新计算各聚类中心，称为成批处理法(本书采用此法)。

(2) 选取一批代表点后，依次计算其他样本的归类。当计算完第 1 个样本时，把它归于最近的一类，形成新的分类。然后计算新的聚类中心，再计算第 2 个样本到新的聚类中心的

距离，对第 2 个样本进行归类，即每个样本的归类都改变 1 次聚类中心。此法称为逐个处理法。

（3）直接用样本进行初始分类，先规定距离 d，把第 1 个样本作为第一类的聚类中心。考察第 2 个样本，若第 2 个样本距第一个聚类中心的距离小于 d，就把第 2 个样本归于第一类；否则，第 2 个样本就成为第二类的聚类中心。再考虑其他样本，根据样本到聚类中心的距离大于还是小于 d，决定是分裂还是合并。

（4）最佳初始分类。

如图 4-2 所示，随着初始分类 K 的增大，准则函数下降很快，经过拐点 A 后，下降速度减慢。拐点 A 就是最佳初始分类。

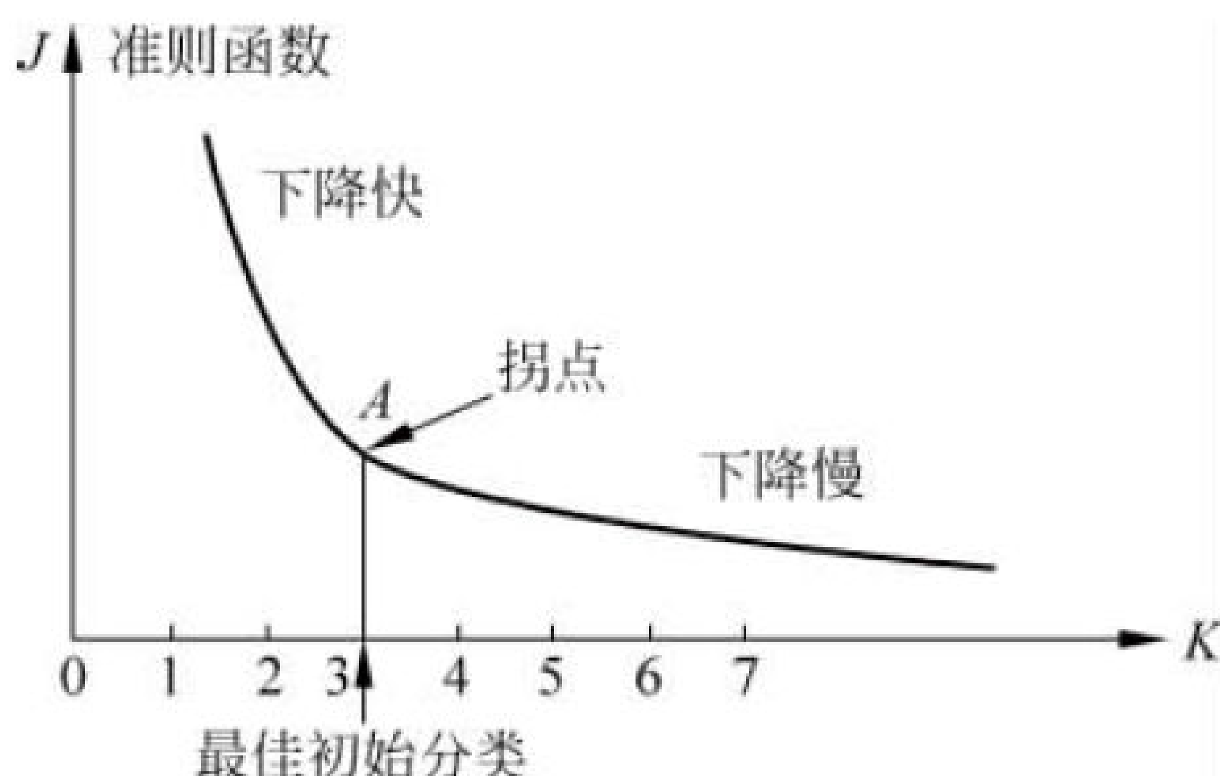


图 4-2 最佳初始分类

4.2.3 K 均值算法的优缺点

K 均值聚类算法在实际生活中非常实用，也非常方便，它不仅能够使得烦琐的数据在计算当中简单化，而且使用范围也比较广。例如，它在对交通事故多发地带的分析方面尤为突出。下面是关于 K 均值算法的几个优点：

（1）如果变量很大，K 均值比层次聚类的计算速度更快。

（2）与层次聚类相比，K 均值可以得到更紧密的簇，尤其是对于球状簇。

（3）大数据集合，效率比较高。

（4）算法尝试找出使平方误差函数最小的 K 个划分。当结果簇是密集的，而簇与簇之间区别明显的时候，效果较好。

任何事情都不是完美的，虽然 K 均值算法在实际生活中非常实用，但是它也有不足的一面。下面是关于 K 均值算法的几点不足：

（1）没有指明初始化均值的方法。常用的方法是随机地选取 K 个样本作为均值。

（2）产生的结果依赖于均值的初始值，经常发生得到次优划分的情况。解决方法是多次尝试不同的初始值。

（3）可能发生距离簇中心 m_j 最近的样本集为空的情况，因此 m_j 将得不到更新。

（4）不适合发现非凸面形状的簇，并且对噪声和离群点数据是比较敏感的，因为少量的这类数据能够对均值产生极大的影响。

4.2.4 K 均值聚类的 MATLAB 实现

以表 1-2 所示的样本数据为例，说明 K 均值聚类的 MATLAB 实现。其中，前 29 组数据已确定类别；后 30 组数据为待确定类别。

在 MATLAB 中，直接调用如下程序即可实现 K 均值聚类。

```
[IDX,C,SUMD,D] = kmeans(data,K);
```

其中，data：要聚类的数据集合，每一行为一个样本；IDX：聚类结果；C：聚类中心；SUMD：每一个样本到该聚类中心的距离和；D：每一个样本到各个聚类中心的距离；K：分类的个数。

如果使用命令[IDX,C,SUMD,D] = kmeans(data,4)进行聚类，要想画出 4 个聚类的图形，可用如下程序代码：

```
D = D          % 得到每一个样本到四个聚类中心的距离
minD = min(D); % 找到每一个样本到四个聚类中心的最小距离
index1 = find(D(1,:) == min(D)) % 找到属于第一类的点
index2 = find(D(2,:) == min(D)) % 找到属于第二类的点
index3 = find(D(3,:) == min(D)) % 找到属于第三类的点
index4 = find(D(4,:) == min(D)) % 找到属于第四类的点
```

为了提高图形的区分度，添加如下命令：

```
line(data(index1,1),data(index1,2),data(index1,3),'linestyle','none','marker','*','color','g');
line(data(index2,1),data(index2,2),data(index2,3),'linestyle','none','marker','*','color','r');
line(data(index3,1),data(index3,2),data(index3,3),'linestyle','none','marker','+','color','b');
line(data(index4,1),data(index4,2),data(index4,3),'linestyle','none','marker','+','color','y');
```

(1) 初始分类的选取和调整。

K 均值算法的类型数目假定已知为 K。对于 K 未知时，可以令 K 逐渐增加。使用 K 均值算法，误差平方和 J_K 随 K 的增加而单调减少。最初，由于 K 较小，类型的分裂会使 J_K 迅速减小，但当 K 增加到一定数值时，J_K 的减小速度会减慢，即随着初始分类 K 的增大，准则函数下降很快，经过拐点后，下降速度减慢。拐点处的 K 值就是最佳初始分类。

(2) 当分类的数目 $K=2$ 时，调用如下程序实现 K 均值聚类：

```
[IDX,C,SUMD,D] = kmeans(data,2);
SUMD =
  1.0e+007 *
    1.4810
    1.0599
J_K = 25.409 * 106
```

(3) 当分类的数目 $K=3$ 时,调用如下程序实现 K 均值聚类:

```
[IDX,C,SUMD,D] = kmeans(data,3);
SUMD =
  1.0e+006 *
    1.4058
    0.3332
    7.3544
JK = 9.0934 * 106
```

(4) 当分类的数目 $K=4$ 时,调用如下程序实现 K 均值聚类:

```
[IDX,C,SUMD,D] = kmeans(data,4);
SUMD =
1.0e+006 *
    1.1820
    1.5262
    0.3332
    1.4058
JK = 4.4472 * 106
```

(5) 当分类的数目 $K=5$ 时,调用如下程序实现 K 均值聚类:

```
[IDX,C,SUMD,D] = kmeans(data,5);
SUMD =
  1.0e+006 *
    0.3332
    0.0437
    0.8008
    0.9260
    1.4058
JK = 3.5095 * 106
```

(6) 当分类的数目 $K=6$ 时,调用如下程序实现 K 均值聚类:

```
[IDX,C,SUMD,D] = kmeans(data,6);
SUMD =
  1.0e+006 *
    0.4764
    1.2492
    0.3223
    0.2014
    0.5362
    0.3332
JK = 3.1187 * 106
```

(7) 当分类的数目 $K=7$ 时,调用如下程序实现 K 均值聚类:

```
[IDX,C,SUMD,D] = kmeans(data,7);
SUMD =
  1.0e+005 *
    3.2850
    2.0138
    2.7520
    3.1015
    5.3616
    3.7949
    3.3317
JK = 2.3641 * 106
```

如图 4-3 所示,随着初始分类 K 的增大,准则函数下降很快,经过拐点后,下降速度减慢。拐点就是最佳初始分类,即 $K=4$ 时为最佳初始分类。

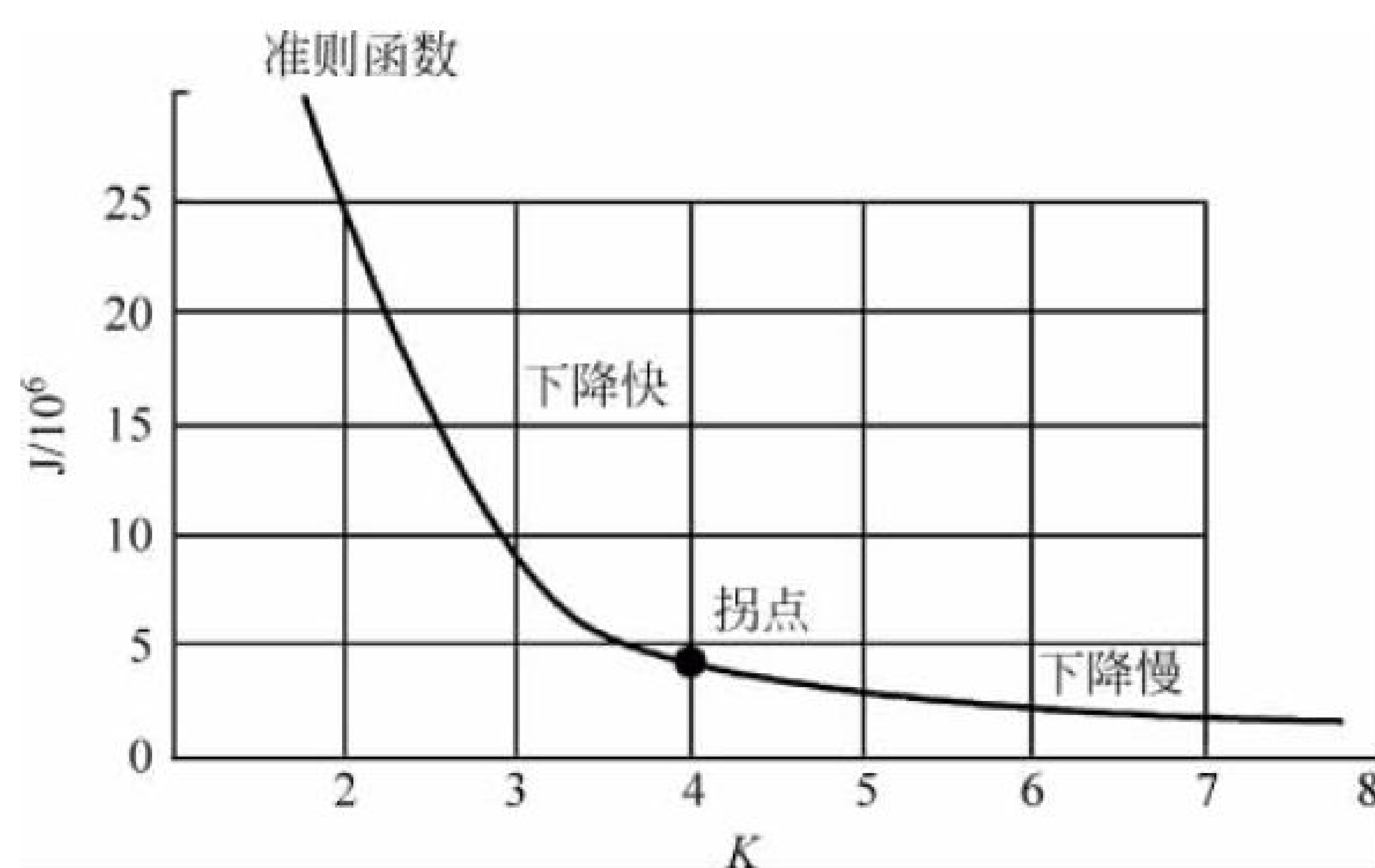


图 4-3 最佳初始分类图

完整分类的 MATLAB 源程序代码如下:

```
clear all;
data = [
1702.8     1639.79     2068.74
1877.93    1860.96     1975.3
867.81     2334.68     2535.1
1831.49    1713.11     1604.68
460.69     3274.77     2172.99
2374.98    3346.98     975.31
2271.89    3482.97     946.7
1783.64    1597.99     2261.31
198.83     3250.45     2445.08
1494.63    2072.59     2550.51
1597.03    1921.52     2126.76
1598.93    1921.08     1623.33
1243.13    1814.07     3441.07
2336.31    2640.26     1599.63
354        3300.12     2373.61
```

```
2144.47     2501.62     591.51
426.31      3105.29     2057.8
1507.13     1556.89     1954.51
343.07      3271.72     2036.94
2201.94     3196.22     935.53
2232.43     3077.87     1298.87
1580.1      1752.07     2463.04
1962.4      1594.97     1835.95
1495.18     1957.44     3498.02
1125.17     1594.39     2937.73
24.22       3447.31     2145.01
1269.07     1910.72     2701.97
1802.07     1725.81     1966.35
1817.36     1927.4      2328.79
1860.45     1782.88     1875.13
];
[IDX,C,SUMD,D] = kmeans(data,4);
plot3(data(:,1),data(:,2),data(:,3),'*');
grid;
D = D'
minD = min(D);
index1 = find(D(1,:) == min(D))
index2 = find(D(2,:) == min(D))
index3 = find(D(3,:) == min(D))
index4 = find(D(4,:) == min(D))
line(data(index1,1),data(index1,2),data(index1,3),'linestyle', 'none','marker','*','color','g');
line(data(index2,1),data(index2,2),data(index2,3),'linestyle', 'none','marker','*','color','r');
line(data(index3,1),data(index3,2),data(index3,3),'linestyle', 'none','marker','+','color','b');
line(data(index4,1),data(index4,2),data(index4,3),'linestyle', 'none','marker','+','color','y');
title('K均值聚类分类图');
xlabel('第一特征坐标');
ylabel('第二特征坐标');
zlabel('第三特征坐标');
```

4.2.5 待聚类样本的分类结果

待聚类样本按下述步骤分类。

(1) 所分4类的聚类中心C,实现代码如下:

```
C =
  1.0e+03 *
    1.2964    1.9194    2.8753(index1 聚类中心)
    0.3012    3.2749    2.2052(index2 聚类中心)
    2.2603    3.0410    1.0579(index3 聚类中心)
    1.7583    1.7493    1.9655(index4 聚类中心)
```

(2) 所分的4类,实现代码如下:

```
index1 =
    3    10    13    22    24    25    27
index2 =
    5     9    15    17    19    26
index3 =
    6     7    14    16    20    21
index4 =
    1     2     4     8    11    12    18    23    28    29    30
```

执行上述代码后，待分类样本 K 均值聚类 MATLAB 分类图界面如图 4-4 所示。

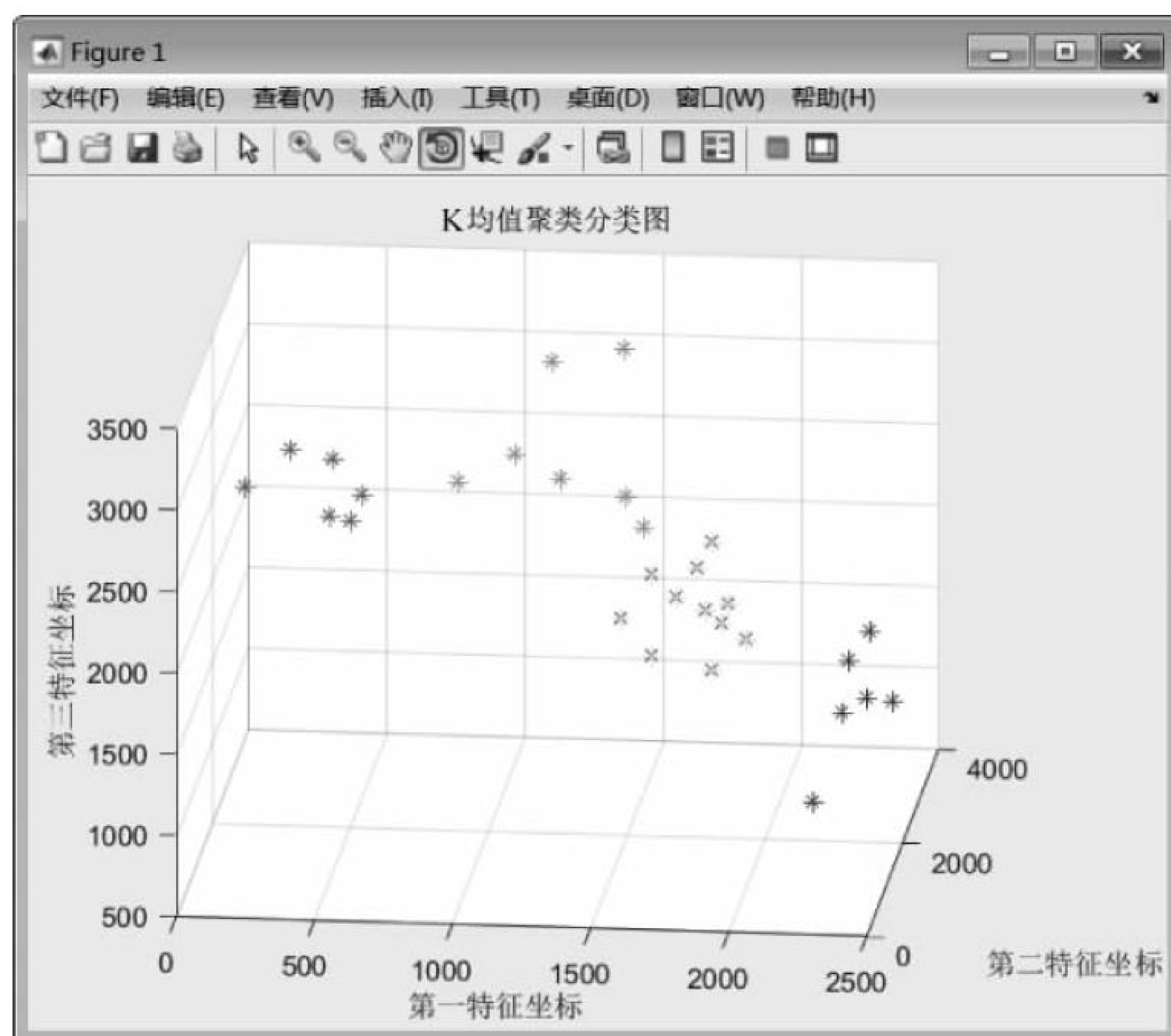


图 4-4　待分类样本 K 均值聚类 MATLAB 分类图界面

4.2.6　结论

对 30 个样本进行分类的 MATLAB 程序代码运行结果如下：

```
D =
  1.0e+06 *
  1～8 列
    0.8939    1.1516    0.4719    1.9434    3.0288    6.8113    7.1159    0.7177
    4.6569    4.5383    1.3140    5.1418    0.0265    5.8185    5.5109    5.0130
    3.2959    2.3803    4.6200    2.2461    4.5368    0.1136    0.2079    3.7576
    0.0257    0.0269    1.4600    0.1369    4.0538    3.9134    4.3074    0.1110
  9～16 列
    3.1615    0.1683    0.6507    1.6591    0.3340    3.2284    3.0463    6.2740
```

```
   0.0686    2.9892    3.5171    3.8557    4.5487    4.9113    0.0318    6.5998
   6.2179    3.7519    2.8356    2.0113    8.2194    0.4598    5.4323    0.5219
   4.9153    0.5162    0.0817    0.1720    2.4468    1.2618    4.5436    2.6030
 17～24 列
   2.8318    1.0238    3.4406    6.2130    4.7034    0.2785    1.6291    0.4287
   0.0662    4.4689    0.0301    5.2312    4.5900    4.0212    5.7183    4.8327
   4.3675    3.5737    4.6876    0.0425    0.0602    4.0984    2.7851    7.7136
   3.6214    0.1002    4.3257    3.3513    2.4343    0.2793    0.0823    2.4611
 25～30 列
   0.1389    4.4864    0.0309    1.1194    0.5701    1.3372
   4.0398    0.1100    3.0446    4.7095    4.1299    4.7665
   6.9149    6.3471    4.9630    2.7649    3.0514    2.4106
   1.3700    5.9224    0.8077    0.0025    0.1672    0.0197
```

其中，D 为每个样本与聚合中心的最小距离。

分类结果如下：

```
index1 =
    3    10    13    22    24    25    27
index2 =
    5     9    15    17    19    26
index3 =
    6     7    14    16    20    21
index4 =
    1     2     4     8    11    12    18    23    28    29    30
```

通过对分类结果的验证表明，该算法能很好地对样本进行分类。使用该算法得到的结果显示全部样本分类结果与正确样本的分类结果完全符合。

在 K 均值聚类算法中，Kmeans 算法主要通过迭代搜索获得聚类的划分结果。虽然 Kmeans 算法运算速度快，占用内存小，比较适合于大样本量的情况，但是聚类结果受初始凝聚点的影响很大，不同的初始点选择会导致截然不同的结果。并且当按最近邻归类时，如果遇到与两个聚类中心距离相等的情况，不同的选择也会造成不同的结果。因此，K 均值动态聚类法具有因初始聚类中心的不确定性而存在较大偏差的情况。

K 均值算法使用的聚类准则函数是误差平方和准则。在算法迭代过程中，样本分类不断调整，因此误差平方和 J_K 也在逐步减小，直到没有样本调整为止，此时 J_K 不再变化，聚类达到最优。但是，此算法中没有计算 J_K 值，也就是说 J_K 不是算法结束的明显依据。因此，有待进一步对 K 均值算法进行改进，以优化 K 均值聚类算法。

4.3 数据聚类——基于取样思想的改进 K 均值聚类

K 均值算法属于聚类技术中的一种基本划分方法，具有简单、快速的优点，但也存在因初始聚类中心不确定性而存在较大偏差的情况。为此对 K 均值算法的初始聚类中心选择方法进行了改进，提出了一种从数据对象分布出发动态寻找并确定初始聚类中心的思路以

及基于这种思路的改进算法。

4.3.1 K 均值改进算法的思想

在 K 均值算法中，选择不同的初始聚类中心会产生不同的聚类结果且有不同的准确率，此方法就是如何找到与数据在空间分布上尽可能一致的初始聚类中心。对数据进行划分，最根本的目的是使一个聚类中的对象是相似的，而不同聚类中的对象是不相似的。如果用距离表示对象之间的相似性程度，相似对象之间的距离比不相似对象之间的距离要小。如果能够寻找到 K 个初始中心，它们分别代表了相似程度较大的数据集合，那么就找到了与数据在空间分布上相一致的初始聚类中心。

目前，选取初始聚类中心的方法有很多种，在此仅介绍两种。

1. 基于最小距离的初始聚类中心选取法

(1) 计算数据对象两两之间的距离。

(2) 找出距离最近的两个数据对象，形成一个数据对象集合 A_1，并将它们从总的数据集合 U 中删除。

(3) 计算 A_1 中每一个数据对象与数据对象集合 U 中每一个样本的距离，找出在 U 中与 A_1 中最近的数据对象，将它并入集合 A_1 并从 U 中删除，直到 A_1 中的数据对象个数到达一定阈值。

(4) 再从 U 中找到样本两两间距离最近的两个数据对象构成 A_2，重复上面的过程，直到形成 k 个对象集合。

(5) 最后对 k 个对象集合分别进行算术平均，形成 k 个初始聚类中心。

这种方法和 Huffman 算法一样。

2. 基于最小二叉树的方法

(1) 计算任意两个数据对象间的距离 $d(x,y)$，找到集合 U 中距离最近的两个数据对象，形成集合 $A_m(1\leqslant m\leqslant k)$，并从集合 U 中删除这两个对象。

(2) 在 U 中找到距离集合 A_m 最近的数据对象，将其加入集合 A_m，并从集合 U 中删除该对象。

(3) 重复步骤(2)，直到集合中的数据对象个数大于等于 $a\times\frac{n}{k}(0<a\leqslant1)$。

(4) 如果 $m<k$，则 $m=m+1$，再从集合 U 中找到距离最近的两个数据对象，形成新的集合 $A_m(1\leqslant m\leqslant k)$，并从集合 U 中删除这两个数据对象，返回步骤(2)执行。

(5) 将最终形成的 k 个集合中的数据对象分别进行算术平均，从而形成 k 个初始聚类中心。

说明：$a\times\frac{n}{k}$ 的取值会因实验数据的不同而有所不同。$a\times\frac{n}{k}$ 的取值过小，将使几个初始聚类中心点聚集在同一区域；$a\times\frac{n}{k}$ 的取值过大，又会使初始聚类中心点偏离密集区域。

所以，阈值 $a\times\frac{n}{k}$需要从多次实验中获取。

从这 k 个初始聚类中心出发，应用 K 均值聚类算法形成最终聚类。

4.3.2 基于取样思想的改进 K 均值算法 MATLAB 实现

首先对样本数据采用 K 均值算法进行聚类，产生一组聚类中心，然后将这组聚类中心作为初始聚类中心，再采用 K 均值算法进行聚类。

在此，也可以在第一步中，对样本数据采用 K 均值算法进行 n 次聚类运算，每次产生一组聚类中心，对 n 个聚类中心进行算术平均，从而得到 k 个初始聚类中心。

确定初始聚类中心的 MATLAB 程序如下：

```
[IDX,C] = kmeans(data,k)
```

其中，IDX：聚类结果；C：聚类中心；k：分类个数；data：要聚类的数据集合，每一行为一个样本。

运行 MATLAB 程序代码后，得到的初始聚类中心如下：

```
C =
    1.0e+003 *
    2.3327    3.0789    1.0759
    1.7332    1.7356    1.9762
    1.2106    1.8780    2.9579
    0.3010    3.2228    2.2502
```

基于取样思想的改进 K 均值算法程序如下：

```
function yy = kmeans2()
data = [1739.94  1675.15  2395.96     %样本空间
       373.3    3087.05  2429.47
       1756.77  1652     1514.98
       864.45   1647.31  2665.9
       222.85   3059.54  2002.33
       877.88   2031.66  3071.18
       1803.58  1583.12  2163.05
       2352.12  2557.04  1411.53
       401.3    3259.94  2150.98
       363.34   3477.95  2462.86
       1571.17  1731.04  1735.33
       104.8    3389.83  2421.83
       499.85   3305.75  2196.22
       2297.28  3340.14  535.62
       2092.62  3177.21  584.32
       1418.79  1775.89  2772.9
       1845.59  1918.81  2226.49
       2205.36  3243.74  1202.69
```

```
        2949.16  3244.44  662.42
        1692.62  1867.5   2108.97
        1680.67  1575.78  1725.1
        2802.88  3017.11  1984.98
        172.78   3084.49  2328.65
        2063.54  3199.76  1257.21
        1449.58  1641.58  3405.12
        1651.52  1713.28  1570.38
        341.59   3076.62  2438.63
        291.02   3095.68  2088.95
        237.63   3077.78  2251.96
        1702.8   1639.79  2068.74
        1877.93  1860.96  1975.3
        867.81   2334.68  2535.1
        1831.49  1713.11  1604.68
        460.69   3274.77  2172.99
        2374.98  3346.98  975.31
        2271.89  3482.97  946.7
        1783.64  1597.99  2261.31
        198.83   3250.45  2445.08
        1494.63  2072.59  2550.51
        1597.03  1921.52  2126.76
        1598.93  1921.08  1623.33
        1243.13  1814.07  3441.07
        2336.31  2640.26  1599.63
        354      3300.12  2373.61
        2144.47  2501.62  591.51
        426.31   3105.29  2057.8
        1507.13  1556.89  1954.51
        343.07   3271.72  2036.94
        2201.94  3196.22  935.53
        2232.43  3077.87  1298.87
        1580.1   1752.07  2463.04
        1962.4   1594.97  1835.95
        1495.18  1957.44  3498.02
        1125.17  1594.39  2937.73
        24.22    3447.31  2145.01
        1269.07  1910.72  2701.97
        1802.07  1725.81  1966.35
        1817.36  1927.4   2328.79
        1860.45  1782.88  1875.13
        ];
[IDX,C] = kmeans(data,4);
C
y = [1:59];
z = [data,IDX]';
x = [z;y];
x1 = [];x2 = [];x3 = [];x4 = [];
for i = 1:59
    if x(4,i) == 1
```

```
        x1 = [x1,x(:,i)];
    elseif x(4,i) == 2
        x2 = [x2,x(:,i)];
    elseif x(4,i) == 3
        x3 = [x3,x(:,i)];
    else  x(4,i) == 4
        x4 = [x4,x(:,i)];
    end
end
format short g
x1 = C(1,:)';
x2 = C(2,:)';
x3 = C(3,:)';
x4 = C(4,:)';
x = [x(1:3,:);x(5,:)];
xx = [mean(x1,2),mean(x2,2),mean(x3,2),mean(x4,2)];
xxx = ones(3,4);
j = 0;
z
while xx~ = xxx
    xx = xxx;
    d1 = []; d2 = []; d3 = []; d4 = [];
    for i = 1:size(z,2) d1 = [d1,round(1000 * sum((x(1:3,i) - mean(x1,2)).^2))/1000];
        d2 = [d2,round(1000 * sum((x(1:3,i) - mean(x2,2)).^2))/1000];
        d3 = [d3,round(1000 * sum((x(1:3,i) - mean(x3,2)).^2))/1000];
        d4 = [d4,round(1000 * sum((x(1:3,i) - mean(x4,2)).^2))/1000];
    end
    d1,d2,d3,d4
ww1 = [];ww2 = [];ww3 = [];ww4 = [];
for i = 1:size(z,2)
    if min([d1(i),d2(i),d3(i),d4(i)]) == d1(i)
        ww1 = [ww1,x(:,i)];
    elseif min([d1(i),d2(i),d3(i),d4(i)]) == d2(i)
        ww2 = [ww2,x(:,i)];
    elseif min([d1(i),d2(i),d3(i),d4(i)]) == d3(i)
        ww3 = [ww3,x(:,i)];
    else
        ww4 = [ww4,x(:,i)];
    end
end
x1 = ww1(1:3,:);
x2 = ww2(1:3,:);
x3 = ww3(1:3,:);
x4 = ww4(1:3,:);
xxx = [mean(x1,2),mean(x2,2),mean(x3,2),mean(x4,2)]
yyy = xxx'
end
ww1
ww2
ww3
ww4
plot3(ww1(1,:),ww1(2,:),ww1(3,:),'s',ww2(1,:),ww2(2,:),ww2(3,:),' * ',ww3(1,:),ww3(2,:),
ww3(3,:),'o',ww4(1,:),ww4(2,:),ww4(3,:),' * ')
grid
```

运行 MATLAB 程序代码后，结果如下：

```
C =
    1.0e+03 *
    1.2106  1.8780  2.9579
    2.3327  3.0789  1.0759
    0.3010  3.2228  2.2502
    1.7332  1.7356  1.9762
ww1 =
  1～6 列
              864.45    877.88    1418.8    1449.6    867.81    1494.6
              1647.3    2031.7    1775.9    1641.6    2334.7    2072.6
              2665.9    3071.2    2772.9    3405.1    2535.1    2550.5
                   4         6        16        25        32        39
  7～10 列
              1243.1    1495.2    1125.2    1269.1
              1814.1    1957.4    1594.4    1910.7
              3441.1      3498    2937.7      2702
                  42        53        54        56
ww2 =
  1～6 列
              2352.1    2297.3    2092.6    2205.4    2949.2    2802.9
                2557    3340.1    3177.2    3243.7    3244.4    3017.1
              1411.5    535.62    584.32    1202.7    662.42      1985
                   8        14        15        18        19        22
  7～12 列
              2063.5      2375    2271.9    2336.3    2144.5    2201.9
              3199.8      3347      3483    2640.3    2501.6    3196.2
              1257.2    975.31     946.7    1599.6    591.51    935.53
                  24        35        36        43        45        49
  13 列
              2232.4
              3077.9
              1298.9
                  50
ww3 =
  1～6 列
                373.3    222.85     401.3    363.34     104.8    499.85
               3087.1    3059.5    3259.9    3477.9    3389.8    3305.8
               2429.5    2002.3      2151    2462.9    2421.8    2196.2
                    2         5         9        10        12        13
  7～12 列
              172.78    341.59    291.02    237.63    460.69    198.83
              3084.5    3076.6    3095.7    3077.8    3274.8    3250.4
              2328.7    2438.6    2088.9      2252      2173    2445.1
                  23        27        28        29        34        38
  13～16 列
                 354    426.31    343.07     24.22
              3300.1    3105.3    3271.7    3447.3
              2373.6    2057.8    2036.9      2145
                  44        46        48        55
```

```
ww4 =
  1～6 列
              1739.9    1756.8    1803.6    1571.2    1845.6    1692.6
              1675.2      1652    1583.1      1731    1918.8    1867.5
                2396      1515    2163.1    1735.3    2226.5      2109
                   1         3         7        11        17        20
  7～12 列
              1680.7    1651.5    1702.8    1877.9    1831.5    1783.6
              1575.8    1713.3    1639.8      1861    1713.1      1598
              1725.1    1570.4    2068.7    1975.3    1604.7    2261.3
                  21        26        30        31        33        37
  13～18 列
                1597    1598.9    1507.1    1580.1    1962.4    1802.1
              1921.5    1921.1    1556.9    1752.1      1595    1725.8
              2126.8    1623.3    1954.5      2463      1836    1966.3
                  40        41        47        51        52        57
  19～20 列
              1817.4    1860.5
              1927.4    1782.9
              2328.8    1875.1
                  58        59
```

分类效果图界面如图 4-5 所示。

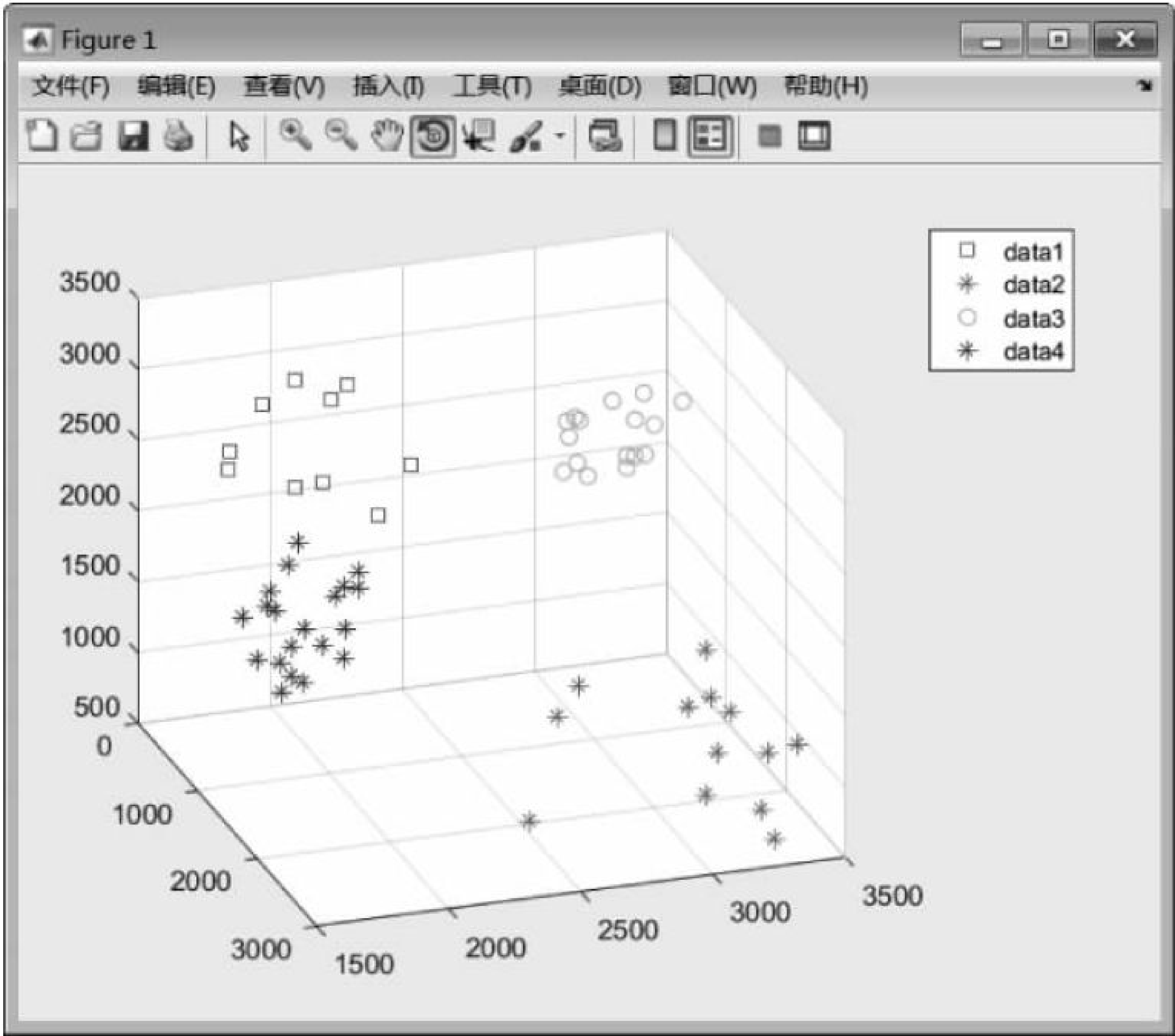


图 4-5 分类效果图界面

4.3.3 结论

本节鉴于初始聚类中心对 K 均值聚类算法的影响，以及 K 均值聚类算法的不足，构造了改进 K 均值的聚类算法。该算法通过两种方法选取初始聚类中心，然后在给定初始聚类中心的基础上再次使用 K 均值聚类算法，从而得出聚类结果。全部样本与已知样本完全符合。

4.4 数据聚类——K-近邻法聚类

4.4.1 K-近邻法简介

K-近邻法分类器是一种在线分类器，即分类的时候，直接从训练样本中找出与测试样本最接近的 K 个样本，以判断测试样本的类属。K-近邻法分类器的可扩展性比较差，因为每判决一个测试样本，都要将其与所有训练样本比较一次，计算距离。但是 K-近邻法分类器对处理与训练样本类似情况的时候的精度比较高。所以在样本比较少而对分类速度要求不高的情况下，可以使用 K-近邻法分类器。同样 K-近邻法分类器也可以应用在只有正例训练样本的情况下。在小规模仿真的时候使用精度较高的 K-近邻法分类器，在大规模仿真和实际网络检验的时候避免使用 K-近邻法分类器。

可以看出，尽管 K-近邻法有其优良品质，但是它的一个致命弱点是需要存储全部训练样本，以及繁重的距离计算量。但以简单的方式降低样本数量，只能使其性能降低，这也是使用者不愿意接受的。

K-近邻法的思想如下：首先，计算新样本与训练样本之间的距离，找到距离最近的 K 个近邻，看 K 个近邻多数属于哪一类，就把新样本分为哪一类，通常 K 是不大于 20 的整数。

例如，图 4-6 中 $K=3$ 时，即选择最近的 3 个点，判断它们的类别，于是可以得出圆形的输入实例应该为三角形；同理，当 $K=5$ 时，结果变成方形，此时测试样本 X 被归类为方形类别。

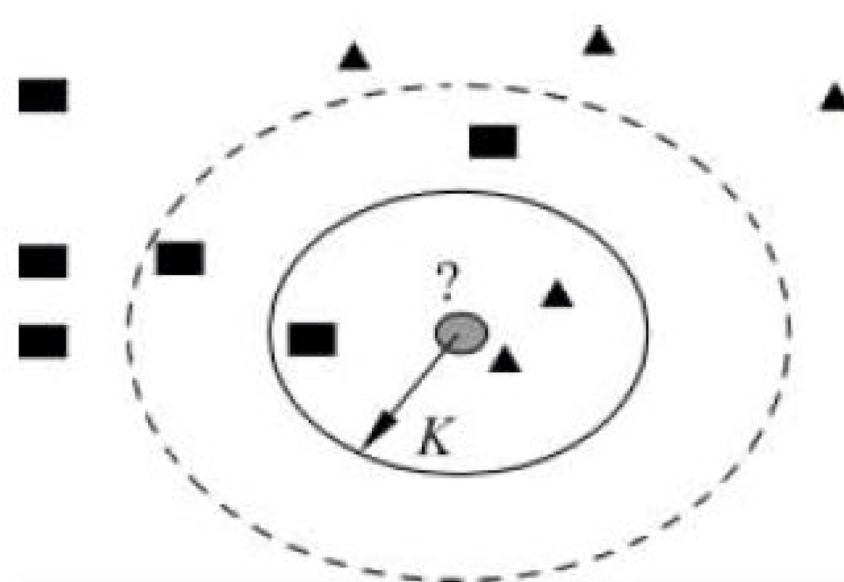


图 4-6 K-近邻法

K-近邻法优点：实现简单，应用范围广，分类效果好。

K-近邻法缺点：样本小时误差难控制，存储所有样本，需要较大存储空间，对于大样本的计算量大。

4.4.2 K-近邻法的算法研究

1. K-近邻法的数学模型

用最近邻方法进行预测的理由是基于假设：近邻的对象具有类似的预测值。最近邻算法的基本思想是在多维空间 R^n 中找到与未知样本最近邻的 K 个点，并根据这 K 个点的类

别来判断未知样本的类。这 K 个点就是未知样本的 K-最近邻。算法假设所有的实例对应于 n 维空间中的点。一个实例的最近邻是根据标准欧氏距离定义，设 x 的特征向量为 $[a_1(x), a_2(x), \cdots, a_n(x)]$。

其中，$a_r(x)$表示实例 x 的第 r 个属性值。两个实例 x_i 和 x_j 间的距离定义为 $d(x_i, x_j)$，其中

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_j))^2} \tag{4-8}$$

在最近邻学习中，离散目标分类函数为 $f: R^n \to V$，其中 V 是有限集合$\{v_1, v_2, \cdots, v_s\}$，即各不同分类集。最近邻数 K 值的选取是根据每类样本中的数目和分散程度进行的，对不同的应用可以选取不同的 K 值。

如果未知样本 s_i 周围的样本点的个数较少，那么 K 个点所覆盖的区域将会很大，反之则小。因此 K-近邻法易受噪声数据的影响，尤其是样本空间中的孤立点的影响。其根源在于基本的 K-最近邻算法中，待预测样本的 K 个最近邻样本的地位是平等的。在自然社会中，通常一个对象受其近邻的影响是不同的，通常是距离越近的对象对其影响越大。

2. K-近邻法的研究方法

该算法没有学习的过程，在分类时通过类别已知的样本对新样本的类别进行预测，因此属于基于实例的推理方法。如果取 $K=1$，待分样本的类别就是最近邻居的类别，称为 KNN 算法。

只要训练样本足够多，K-近邻法就能达到很好的分类效果。当训练样本数趋近于$-\infty$时，K-近邻法的分类误差最差是最优贝叶斯误差的两倍；当 K 趋近于∞时，K-近邻法的分类误差收敛于最优贝叶斯误差。以下是对 K-近邻算法的描述：

(1) 构建训练样本集和测试样本集。

(2) 设定 K 值，一般先确定一个初始值，然后根据实验结果调整到最优。

(3) 计算测试样本和训练样本的欧氏距离。

(4) 选择 K 个近邻的样本，将计算出的距离降序排列，选择距离相对较小的 K 个样本作为测试样本的 K 个近邻。

(5) 找出主要类别，根据 K 个近邻类别，并应用最大概率对所查询的测试样本进行分类。所用概率是指每一个类别出现 K 个近邻的比例，根据每一类别出现在 K 个近邻中的样本数量除以 K 来计算，为 K 个近邻中每一类别样本数量的集合。

(6) 统计 K 个最近邻样本中每个类别出现的次数。

(7) 选择出现频率最高的类别作为未知样本的类别。

输入：训练集和测试集所选的数据集是标准的完整三元色数据。

输出：数据 data 的类别号。

3. K-近邻法需要解决的问题

1) 寻找适当的训练数据集

训练数据集应该是对历史数据的一个很好的覆盖，这样才能保证 K-近邻法有利于预测，选择训练数据集的原则是使各类样本的数量大体一致，另外，选取的历史数据要有代表

性。常用的方法是按照类别把历史数据分组,然后再由每组中选取一些有代表性的样本组成训练集。这样既降低了训练集的大小,又保持了较高的准确度。

2）确定距离函数

距离函数决定了哪些样本是待分样本的 K 个最近邻,它的选取取决于实际的数据和决策问题。如果样本是空间中点,最常用的是欧几里得距离。其他常用的距离函数是绝对距离、平方差和标准差。

3）决定 K 的取值

多数法是最简单的一种综合方法,从邻居中选择一个出现频率最高的类别作为最后的结果,如果频率最高的类别不止一个,就选择最近邻的类别。权重法是较复杂的一种方法,它对 K 个最近邻居设置权重,距离越大,权重就越小。在统计类别时,计算每个类别的权重和,最大的那个就是新样本的类别。

4）K 值的选取

如果 K 值过小,将会对数据中存在的噪声过于敏感;如果 K 值过大,近邻中可能包含属于其他类的样本;一个经验的取值法则为 $K\leqslant\sqrt{q}$,q 为训练样本的数目。商业算法通常以 10 作为默认值。

4.4.3 K-近邻法数据分类器的 MATLAB 实现

以表 1-2 所示数据为例,说明 K 均值聚类的 MATLAB 实现。其中,前 29 组数据已确定类别,后 30 组数据待确定类别。

1. KNN 函数介绍

在 MATLAB 中首先建立 KNN 函数的 m 文件,代码如下:

```
function [label_test] = knn(k,data_train,label_train,data_test)

error(nargchk(4,4,nargin));
%计算出新的特征参数与表 1-2 中参数的距离
dist = l2_distance(data_train,data_test);
%对距离进行排序
[sorted_dist,nearest] = sort(dist);
%选出最近的特征
nearest = nearest(1:k,:);
%用最近的特征的故障类型,作为新的特征参数的故障类型
label_test = label_train(nearest);
```

其中 function [label_test] = knn(k,data_train,label_train,data_test)为定义的 knn 功能函数,它具有 4 个参数,即 K 值、训练数据、训练数据分类及测试数据。该函数的返回值是测试样本的分类结果。

2. 欧氏距离函数

K-近邻法分类器要计算数据与数据的距离,就需要用到欧氏距离函数。该算法定义的

欧氏距离的 m 文件代码如下：

```
function d = l2_distance(X,Y)
% 计算出 x,y 之间的欧氏距离
if (nargin < 2)
    [D N] = size(X);
    lengths = sum(X.^2,1);
    d = repmat(lengths,[N 1]) + repmat(lengths',[1 N]);
    d = d - 2 * X' * X;
    else
    XX = sum(X.^2,1);
    YY = sum(Y.^2,1);
    d = repmat(XX',[1 size(Y,2)]) + repmat(YY,[size(X,2) 1]);
    d = d - 2 * X' * Y;
end
```

3. MATLAB 完整程序

本例 K-近邻法的完整 MATLAB 程序如下：

```
clear;
clc;
DATA = load('D.mat');
%% 绘制训练数据图
first = DATA.train_data(DATA.train_label == 1,:,:);
second = DATA.train_data(DATA.train_label == 2,:,:);
third = DATA.train_data(DATA.train_label == 3,:,:);
fourth = DATA.train_data(DATA.train_label == 4,:,:);
figure;
scatter3(first(:,1),first(:,2),first(:,3),'*');
hold on
scatter3(second(:,1),second(:,2),second(:,3),'p');
scatter3(third(:,1),third(:,2),third(:,3),'s');
scatter3(fourth(:,1),fourth(:,2),fourth(:,3),'o');
title('训练数据');legend('第 1 类','第 2 类','第 3 类','第 4 类');
%% KNN 寻优
acc = zeros(10,1);
for k = 1:10
    % KNN 算法
    label_test = knn(k,DATA.train_data',DATA.train_label',DATA.test_data');
    % 计算最终结果
    if k == 1
        testResults = label_test;
    else
        [maxCount,idx] = max(label_test);
        testResults = maxCount;
```

```
    end
    % 存储各分类结果
    RESULTS(k,:) = testResults;
        % 计算正确率
    count = 0;
    for i=1:30
        if (testResults(i) == DATA.test_label(i))
        count = count+1;
        end
    end
    acc(k) = count/30;
end
disp('精度：')
disp(acc);
%% 求出最优 K
[～,K] = max(acc);
disp('最佳的 K 值为：');
disp(K);
%% 绘制 K=1 时的样本训练数据图,并在命令行窗口显示分类
%% 使用最优 K 进行一次测试
label_test = knn(K,DATA.train_data',DATA.train_label',DATA.test_data');
if K ==1
    testResults = label_test
else
    [maxCount,idx] = max(label_test);
    testResults = maxCount
end
%% 绘制测试数据图
first = DATA.test_data(testResults==1,:,:);
second = DATA.test_data(testResults==2,:,:);
third = DATA.test_data(testResults==3,:,:);
fourth = DATA.test_data(testResults==4,:,:);
figure;
scatter3(first(:,1),first(:,2),first(:,3),'*');
hold on
scatter3(second(:,1),second(:,2),second(:,3),'p');
scatter3(third(:,1),third(:,2),third(:,3),'s');
scatter3(fourth(:,1),fourth(:,2),fourth(:,3),'o');
title('测试数据');legend('第 1 类','第 2 类','第 3 类','第 4 类');
```

单步运行该程序,首先会出现测试样本的分类图页面,如图 4-7 所示。

继续运行程序,将出现测试样本分类结果,如图 4-8 所示。

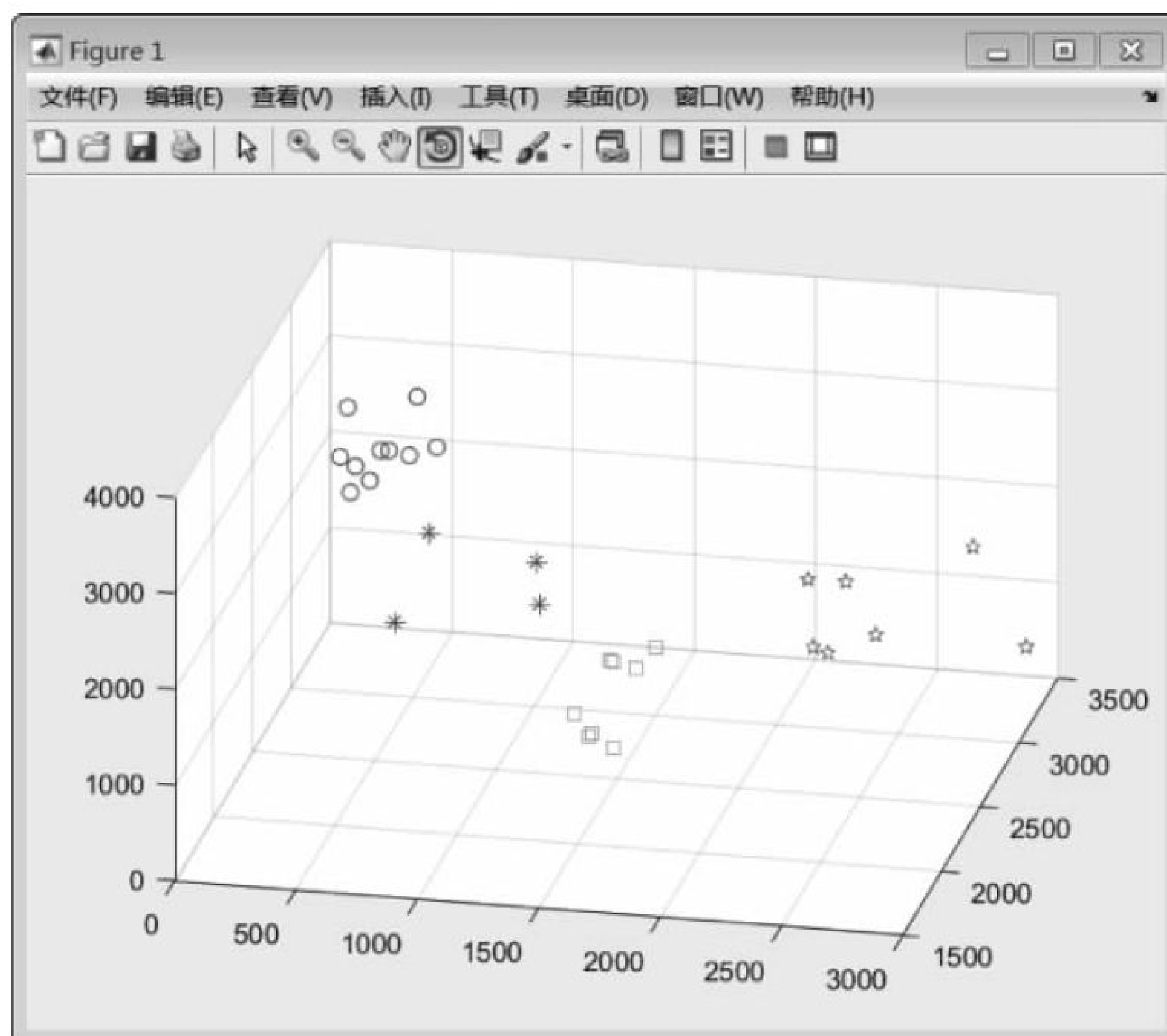


图 4-7 测试样本分类图页面

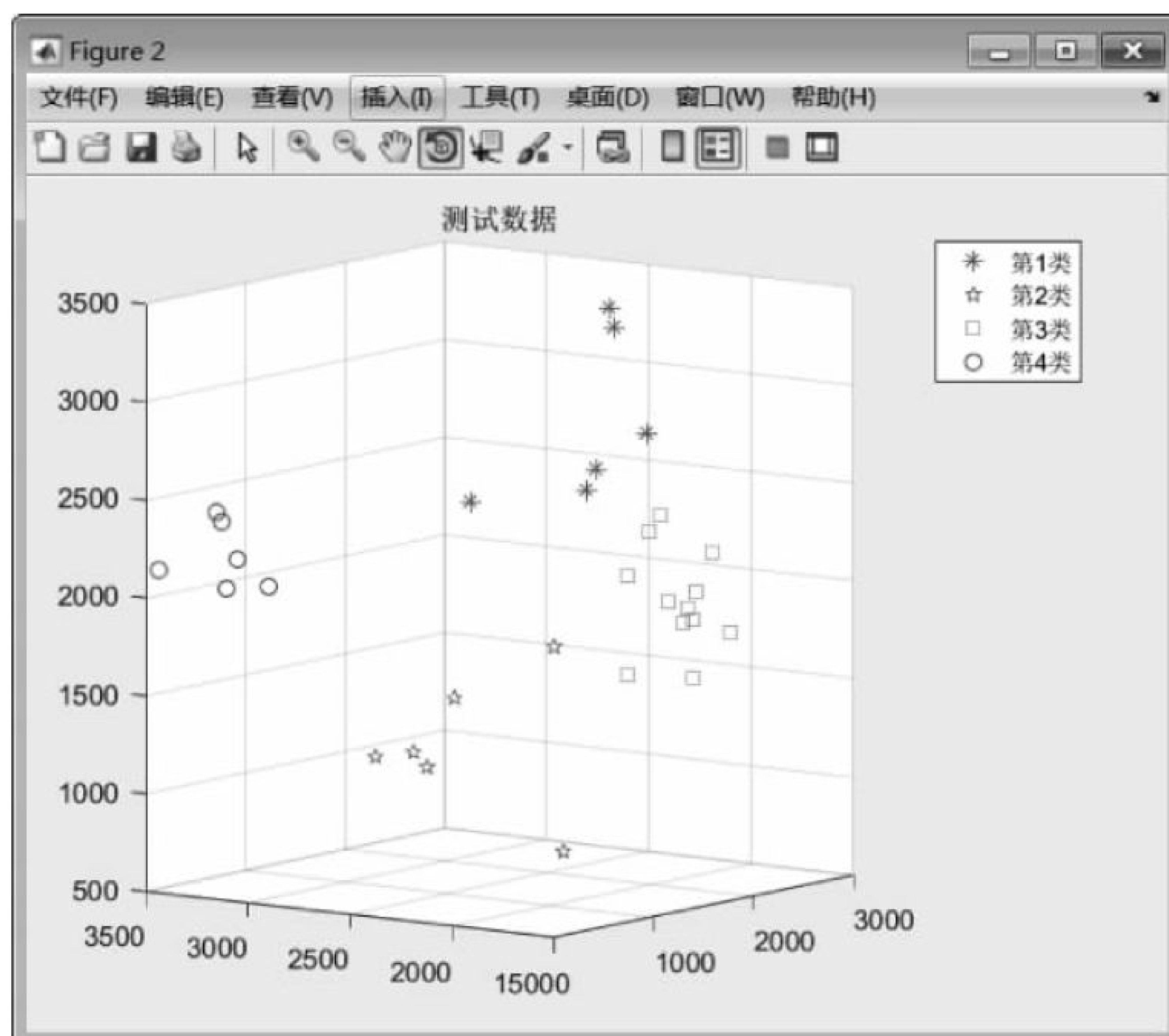


图 4-8 测试样本分类图页面

程序运行完之后,在命令窗口出现如下运行结果:

```
testResults =
1 ~ 15 列
3    3    1    3    4    2    2    3    4    1    3    3    1    2
4
16 ~ 30 列
2    4    3    4    2    2    3    3    1    1    4    1    3    3
3
```

将 KNN 分类器的分类结果与 K 均值分类器的分类结果作比较,如表 4-1 所示。

表 4-1　KNN 分类器的分类结果

序　号	A	B	C	K 均值分类结果	KNN 分类结果
1	1702.8	1639.79	2068.74	3	3
2	1877.93	1860.96	1975.3	3	3
3	867.81	2334.68	2535.1	1	1
4	1831.49	1713.11	1604.68	3	3
5	460.69	3274.77	2172.99	4	4
6	2374.98	3346.98	975.31	2	2
7	2271.89	3482.97	946.7	2	2
8	1783.64	1597.99	2261.31	3	3
9	198.83	3250.45	2445.08	4	4
10	1494.63	2072.59	2550.51	1	1
11	1597.03	1921.52	2126.76	3	3
12	1598.93	1921.08	1623.33	3	3
13	1243.13	1814.07	3441.07	1	1
14	2336.31	2640.26	1599.63	2	2
15	354	3300.12	2373.61	4	4
16	2144.47	2501.62	591.51	2	2
17	426.31	3105.29	2057.8	4	4
18	1507.13	1556.89	1954.51	3	3
19	343.07	3271.72	2036.94	4	4
20	2201.94	3196.22	935.53	2	2
21	2232.43	3077.87	1298.87	2	2
22	1580.1	1752.07	2463.04	1	3
23	1962.4	1594.97	1835.95	3	3
24	1495.18	1957.44	3498.02	1	1
25	1125.17	1594.39	2937.73	1	1
26	24.22	3447.31	2145.01	4	4
27	1269.07	1910.72	2701.97	1	1
28	1802.07	1725.81	1966.35	3	3
29	1817.36	1927.4	2328.79	3	3
30	1860.45	1782.88	1875.13	3	3

从表 4-1 中可以看出，K-近邻法和 K 均值分类仅有一个分类不一致，即(1580.1，1752.07，2463.04)。

4.4.4 结论

本 K-近邻法数据分类器基本实现了数据分类，并且对数据测试结果表明：基本实现了预定目标，达到分类的效果。

K-近邻法具有主观性，因为必须定义一个距离尺度，分类的结果完全依赖使用的距离。这样对于用一组数据，利用 K-近邻法两个不同的分类算法会产生两种基本相同的分类结果，一般需要专家来评测结果是否有效。由于对结果的认识往往是属于经验性的，因此限制了各种距离公式的使用。

4.5 数据聚类——PAM 聚类

4.5.1 PAM 算法简介

围绕中心点的划分(partitioning around medoids，PAM)是聚类分析算法中划分法的一种聚类方法，是最早提出的 K 中心点算法之一。

如今数据挖掘理论越来越广泛地应用在商业、制造业、金融业、医药业、电信业等许多领域。数据挖掘的目标之一是进行聚类分析。聚类就是把一组个体按照相似性归成若干类别，它的目的是使得属于同一类别的个体之间的差别尽可能小，而不同种类别上的个体间的差别尽可能大。PAM 聚类算法是众多聚类算法之一。PAM 算法的优势在于：PAM 算法比 K 均值算法更健壮，对噪声和孤立点数据不敏感；它能够处理不同类型的数据点；它对小的数据集非常有效。

4.5.2 PAM 算法的主要流程

对于 PAM 算法，它的输入是簇的数目 k 和包含 n 个对象的数据库；输出是 k 个簇，使得所有对象与其最近中心点的相异度总和最小。

PAM 算法的主要流程如下：

(1) 任意选择 k 个对象作为初始的簇中心点。

(2) Repeat(重复)。

(3) 指派每个剩余对象给离它最近的中心点所表示的簇。

(4) Repeat。

(5) 选择一个未被选择的中心点 O_i。

(6) Repeat。

(7) 选择一个未被选择过的非中心点对象 O_h。

(8) 计算用 O_h 代替 O_i 的总代价并记录在 S 中。

(9) 直到所有非中心点都被选择过。

(10) 直到所有的中心点都被选择过。

(11) 如果在 S 中的所有非中心点代替所有中心点后的计算出总代价有小于 0 的存在,那么找出 S 中的用非中心点替代中心点后代价最小的一个,并用该非中心点替代对应的中心点,形成一个新的 k 个中心点的集合。

(12) 直到没有再发生簇的重新分配,即所有的 S 都大于 0。

PAM 算法需用簇中位置最靠近中心的对象作为代表对象,然后反复地用非代表对象来代替代表对象,试图找出更好的中心点。在反复迭代的过程中,所有可能的"对象对"被分析,每对中的一个对象是中心点,另一个是非代表对象。一个对象代表可以被最大平方-误差值减少的对象代替。

一个非代表对象 O_h 是否是当前一个代表对象 O_i 的一个好的替代,对于每个非中心点对象 O_j,有以下四种情况需要考虑。

情况一如图 4-9 所示,O_j 当前隶属于 O_i,如果 O_i 被 O_h 替换,且 O_j 离另一个 O_m 最近,那么 O_j 被分配给 O_m,则替换代价为 $C_{jih}=d(j,m)-d(j,i)$。

情况二如图 4-10 所示,O_j 当前隶属于 O_i,如果 O_i 被 O_h 替换,且 O_j 离 O_h 最近,那么 O_j 被分配给 O_h,则替换代价为 $C_{jih}=d(j,h)-d(j,i)$。

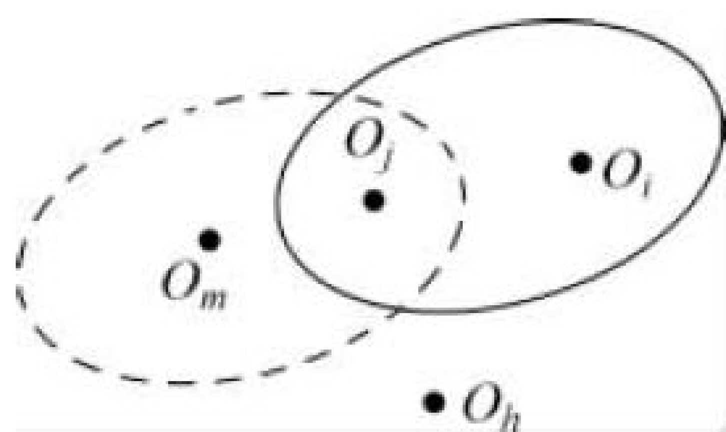


图 4-9 情况一数据分布图

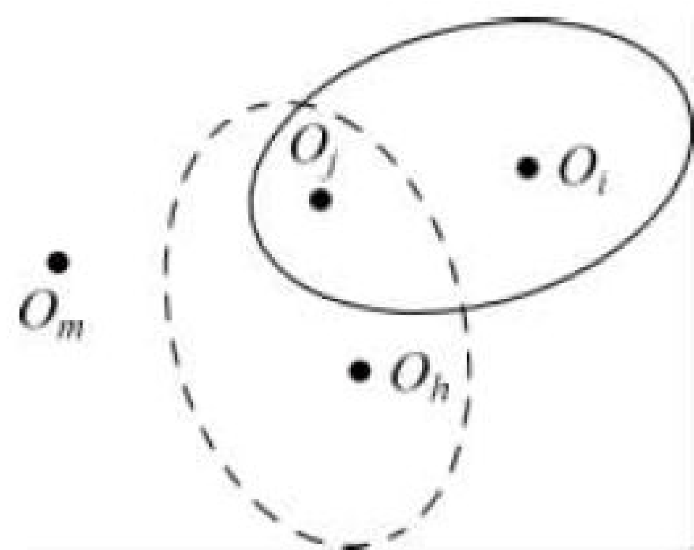


图 4-10 情况二数据分布图

情况三如图 4-11 所示,O_j 当前隶属于 O_m,$m!=i$,如果 O_i 被 O_h 替换,且 O_j 仍然离 O_m 最近,那么 O_j 被分配给 O_m,则替换代价为 $C_{jih}=0$。

情况四如图 4-12 所示,O_j 当前隶属于 O_m,$m!=i$,如果 O_i 被 O_h 替换,且 O_j 离 O_h 最近,那么 O_j 被分配给 O_h,则替换代价为 $C_{jih}=d(j,h)-d(j,m)$。

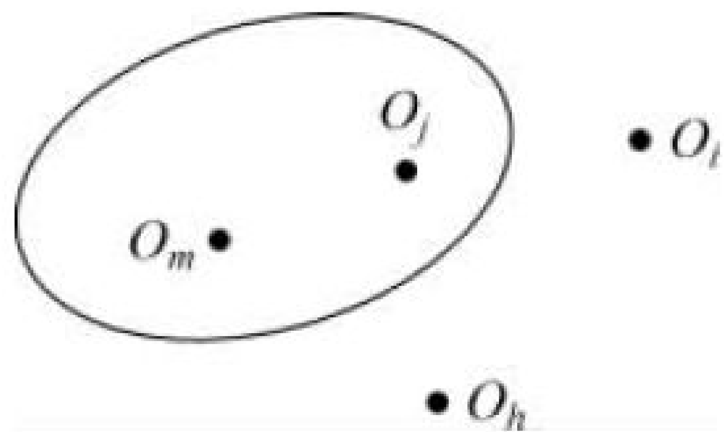


图 4-11 情况三数据分布图

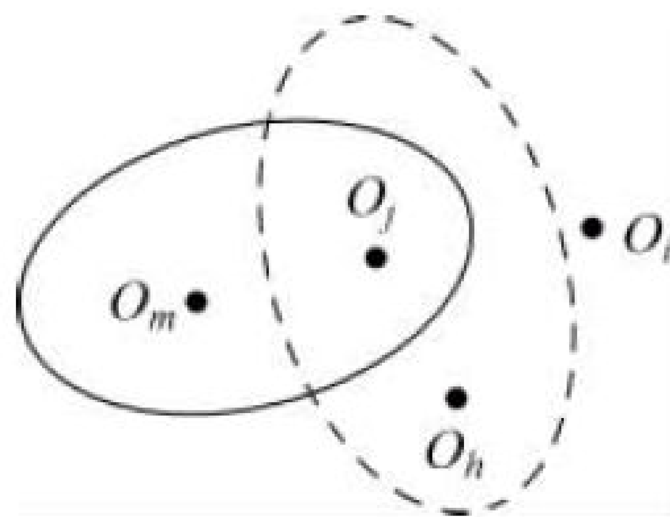


图 4-12 情况四数据分布图

每当重新分配发生时,平方-误差 E 所产生的差别对代价函数有影响。因此,如果一个当前的中心点对象被非中心点对象所代替,则代价函数计算平方-误差值会产生差别,替换的总代价是所有非中心点对象所产生的代价之和。如果总代价是负的,那么实际的平方-误

差将会减小，O_i 可以被 O_h 替代；如果总代价是正的，则当前的中心点 O_i 被认为是可接受的，在本次迭代中没有变化。

4.5.3 PAM 算法的 MATLAB 实现

1. PAM 函数的定义

在 MATLAB 中有 KPAM 的工具箱，通过在 MATLAB 命令窗口输入 help kpam 命令就可以查看 PAM 算法的具体程序。下面仅介绍 KPA 功能函数的直接调用方法，该函数说明如下：

```
%function [result,c,s,index,label] = kpam(data,k)
%result 表示聚类结果
%c 表示聚类最后的中心点
%index 表示随机排列的行号
%label 表示数据每一行属于第几个中心点
%随机选择 k 个中心点
```

2. MATLAB 完整程序

将三元色的后 30 组数据进行聚类，PAM 数据分类的完整程序如下：

```
%function [result,c,s,index,label] = kpam(data,k)
%result 表示聚类结果
%c 表示聚类最后的中心点
%index 表示随机排列的行号
%label 表示数据每一行属于第几个中心点
%随机选择 k 个中心点
clc;clear all;
data = [1702.8    1639.79   2068.74
        1877.93   1860.96   1975.3
        867.81    2334.68   2535.1
        1831.49   1713.11   1604.68
        460.69    3274.77   2172.99
        2374.98   3346.98   975.31
        2271.89   3482.97   946.7
        1783.64   1597.99   2261.31
        198.83    3250.45   2445.08
        1494.63   2072.59   2550.51
        1597.03   1921.52   2126.76
        1598.93   1921.08   1623.33
        1243.13   1814.07   3441.07
        2336.31   2640.26   1599.63
        354       3300.12   2373.61
        2144.47   2501.62   591.51
        426.31    3105.29   2057.8
        1507.13   1556.89   1954.51
```

```
        343.07   3271.72  2036.94
        2201.94  3196.22  935.53
        2232.43  3077.87  1298.87
        1580.1   1752.07  2463.04
        1962.4   1594.97  1835.95
        1495.18  1957.44  3498.02
        1125.17  1594.39  2937.73
        24.22    3447.31  2145.01
        1269.07  1910.72  2701.97
        1802.07  1725.81  1966.35
        1817.36  1926.4   2328.79
        1860.45  1782.88  1875.13];
k = 4;
[N,n] = size(data);
index = randperm(N);                  % 打乱 N 个数的顺序
v = data(index(1:k),:);               % 初始化速度 %
for t = 1:100
 % 指派每个剩余的对象距离它最近的中心点所代表的簇
 % if k == 1
 %           for j = 1:N
 %               label(j) = 1;
 %           end
 %         else
           for i = 1:k
             label(index(i)) = i;
           end
           for j = k + 1:N
               for i = 1:k
               dist(:,i) = sqrt(sum((data(index(j),:) - v(i,:)).^2)); % 计算距离 %
               end
               [m,l] = min(dist');     % 选取距离最小 %
               label(index(j)) = l;
               end
    end
      for i = 1:k
          c(i,:) = v(i,:);
      end
 % 所有非中心点被选择过,所有的中心点被选择过
      for i = 1:k
          for h = k + 1:N
              for j = 1:N
                  c(i,:) = data(index(h),:);
                    dist1 = sqrt(sum((data(j,:) - c(i,:)).^2));             % 计算距离 %
                    for z = 1:k
                          dist2(z) =  sqrt(sum((data(j,:) - c(z,:)).^2));      % 计算距离 %
                    end
                    for y = 1:k
                              dist4(y) = sqrt(sum((data(j,:) - v(y,:)).^2));  % 计算距离 %
```

```
                    end
                        dist3 = sqrt(sum((data(j,:) - v(i,:)).^2));     % 计算距离 %
                    if label(j) == i
                        if dist1 == min(dist2)

                            cjih(j,:) = dist1 - dist3;
                        else
                            cjih(j,:) = min(dist2) - dist3;
                        end
                    else
                        if dist1 == min(dist4)

                            cjih(j,:) = dist1 - min(dist4);
                        else
                            cjih(j,:) = 0;
                        end
                    end
                    c(i,:) = v(i,:);
                    % s1(j,:) = cjih(j,:); ?
                end
 % 一个非中心点代替一个中心点的总代价 s
                        s((h-k),:,i) = sum(cjih(:,:),1);
                    end
                end
              % if min(min(s)) == 0
                  for i = 1:k
                   for h = k + 1:N
                    if s((h-k),:,i) == min(min(s))
                          s((h-k),:,i) = 1;
                    end
                  end
                end
          % end
 % 如果在 S 中的所有非中心点代替所有中心点后计算出的总代价有小于 0 的存在,那么找出 S 中的
 % 用非中心点替代中心点后代价最小的一个,并用该非中心点替代对应的中心点,形成一个新的 k 个
 % 中心点的集合
 if min(min(s))<0
     for i = 1:k
       for h = k + 1:N
         if s((h-k),:,i) == min(min(s))
             v(i,:) = data(index(h),:);
         end
       end
         if data(index(i),:)~ = data(index(h),:)
         if v(i,:) == data(index(h),:)
         end
```

```
            end
        end
        a = index(i);
        b = index(h);
        index(i) = b;
        index(h) = a;
    end
        %所有的 s 都大于 0 则聚类完成
    if min(min(s))>0
    end
 %end
for i = 1:k
for j = 1:N
if label(j) == i
   result(j,:,i) = data(j,:)
% line(result(:,1,1),result(:,2,1),result(:,3,1),'linestyle','none','marker','*','color','g');
% line(result(:,1,2),result(:,2,2),result(:,3,2),'linestyle','none','marker','o','color','b');
% line(result(:,1,3),result(:,2,3),result(:,3,3),'linestyle','none','marker','+','color','r');
% line(result(:,1,4),result(:,2,4),result(:,3,4),'linestyle','none','marker','@','color','y');
% line(data(:,1),data(:,2),data(:,3),'linestyle','none','marker','*','color','r');
% line(data(:,1),data(index3,2),data(index3,3),'linestyle','none','marker','+','color','b');
% line(data(index4,1),data(index4,2),data(index4,3),'linestyle','none','marker','+','color','y');
%title('K均值聚类分类图');
%xlabel('第一特征坐标');
%ylabel('第二特征坐标');
%zlabel('第三特征坐标');
end
end
end
figure
plot3(result(:,1,1),result(:,2,1),result(:,3,1),'*');
hold on;
plot3(result(:,1,2),result(:,2,2),result(:,3,2),'*');
plot3(result(:,1,3),result(:,2,3),result(:,3,3),'*');
plot3(result(:,1,4),result(:,2,4),result(:,3,4),'*');
grid on;
line(result(:,1,1),result(:,2,1),result(:,3,1),'linestyle','none','marker','*','color','y');
line(result(:,1,2),result(:,2,2),result(:,3,2),'linestyle','none','marker','*','color','b');
line(result(:,1,3),result(:,2,3),result(:,3,3),'linestyle','none','marker','*','color','g');
line(result(:,1,4),result(:,2,4),result(:,3,4),'linestyle','none','marker','*','color','r');
```

PAM 算法是通过不断计算中心点及其他点距离中心点的距离来优化分类的，所以需要多次运行程序，找到最优分类。第一次运行程序，将出现如图 4-13 所示的分类结果界面。

图 4-13　首次运行分类结果界面

MATLAB 程序代码运行结果如下：

```
result(:,:,4) =
    1.0e+03 *
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
        2.1445        2.5016        0.5915
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
             0             0             0
```

```
         0            0            0
         0            0            0
         0            0            0
         0            0            0
         0            0            0
         0            0            0
         0            0            0
         0            0            0
```

多次运行之后将出现如图 4-14 所示的分类结果界面。

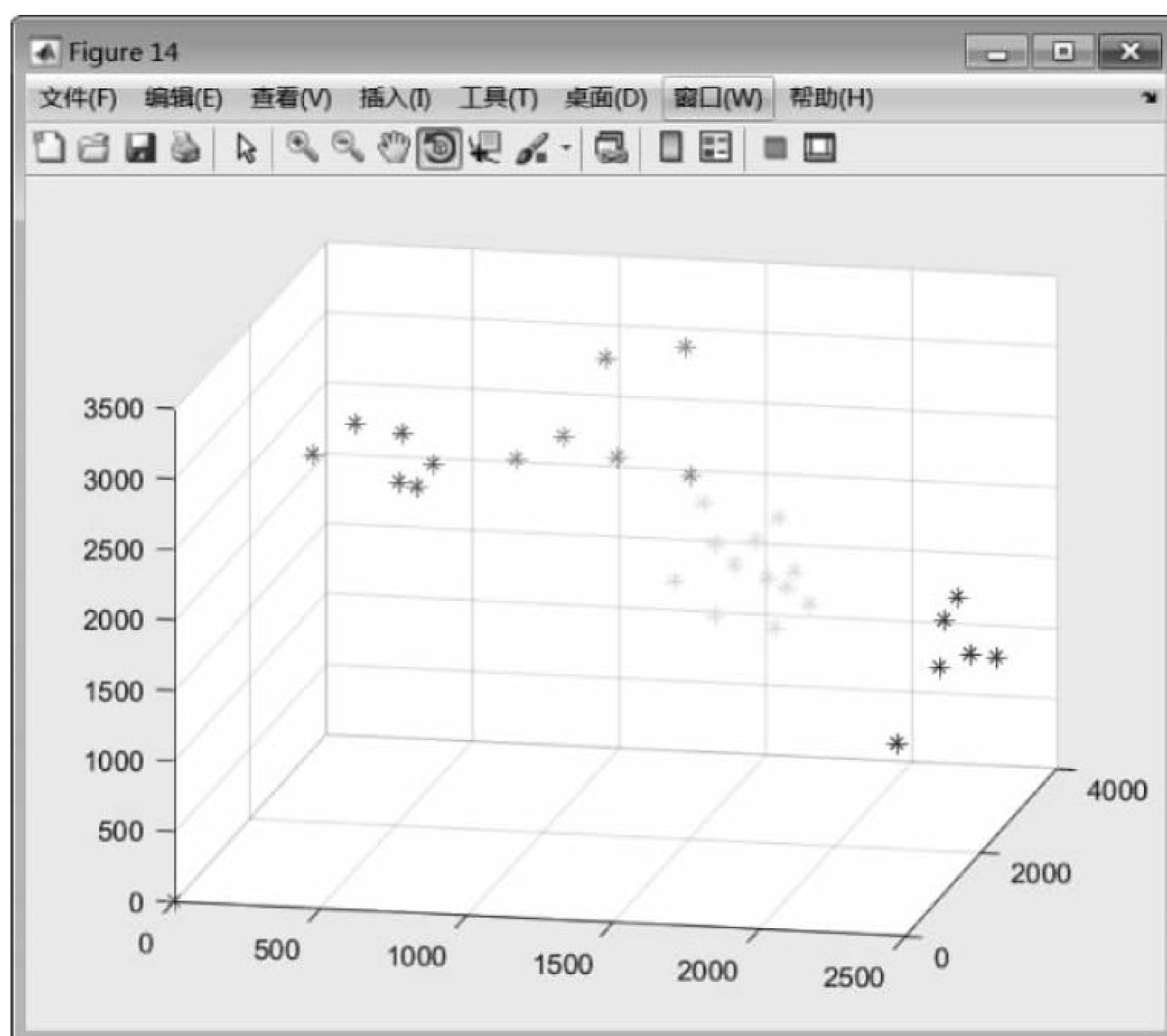


图 4-14 多次运行后的分类结果界面

MATLAB 程序运行完之后出现如下结果：

```
result(:,:,4) =
    1.0e+03 *

         0            0            0
         0            0            0
         0            0            0
         0            0            0
    0.4607       3.2748       2.1730
         0            0            0
         0            0            0
         0            0            0
```

```
    0.1988    3.2504    2.4451
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
    0.3540    3.3001    2.3736
         0         0         0
    0.4263    3.1053    2.0578
         0         0         0
    0.3431    3.2717    2.0369
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
    0.0242    3.4473    2.1450
         0         0         0
         0         0         0
         0         0         0
         0         0         0
```

相关参数的仿真结果会显示在 MATLAB 的工作区中，如图 4-15 所示。

名称	值
a	14
b	20
c	4x3 double
cjih	30x1 double
data	30x3 double
dist	[1.7231e+03,2....
dist1	1.7312e+03
dist2	[128.2032,820....
dist3	1.0185e+03
dist4	[128.2032,820....
h	30
i	4
index	1x30 double
j	30
k	4
l	4
label	1x30 double
m	876.4563
n	3
N	30
result	30x3x4 double
s	26x1x4 double
t	100

图 4-15　MATLAB 程序运行后的工作区

其中 label 项给出了数据的类别号，具体结果如下：

```
     4     4     2     4     1     3     3     4     1     2     4     4     2     3     1     3     1
4     1     3     3     2     4     2     2     1     2     4     4     4
```

通过与标准数据对比，发现 2144.47，2501.62，591.51；2201.94，3196.22，935.53；2232.43，3077.87，1298.87；1962.4，1957.44，3498.02 这 4 组数据的分类不正确，正确率为 87%。这可能是仿真次数不够造成的。

4.5.4 PAM 算法的特点

PAM 算法的一些特定特点，可以帮助用户很好地处理数据问题，但也存在一些不足：

(1) 消除了 K 均值算法对于孤立点的敏感性。

(2) K 中心点方法比 K 均值算法的运算过程复杂，耗时长。

(3) 必须指定 k 的值。

(4) PAM 算法对小的数据集非常有效，对大数据集效率不高。特别是 n 和 k 值都很大的时候。

4.5.5 *K* 均值算法和 PAM 算法分析比较

K 均值算法是先任意选取中心点，通过 k 值将数据分为几类，然后在簇中通过求取平均值的方法重新确定中心点，重新赋值，再重新求取平均值，重复此工作，直至准则 J_K 最小化。而 PAM 算法则是通过对除聚类中心以外的样本点计算到每个聚类中心的距离，再对每个类中除类中心的点外的其他样本点计算到其他所有点的距离和的最小值，将该最小值点作为新的聚类中心便实现了一次聚类优化，也就是样本归类到距离样本中心最近的样本点。这便实现了最初的聚类选取数据中的点作为中心点。检测非中心点 O_j 到中心点的距离与另一个非中心点 O_h 的距离的差是正是负，若为正，则将 O_j 的中心点换为 O_h，依此类推，遍历所有的数据，直至所有非中心点被选择过，所有的中心点被选择过，从而更好地消除孤立点。通过前面的仿真可以知道，PAM 对数据的分类可以减低孤立点的影响(参见图 4-16、图 4-17)。

K 均值算法对孤立点敏感，即使对一个远离聚类中心的目标，算法也强行将其划分一个类中，从而扭曲了聚类的形状。而 PAM 算法以真实数据点为聚类原型，消除了孤立点的影响。

4.5.6 结论

由于现在科学技术的发展，大量的不同数据出现在各行各业，从而使人们处理这些复杂的数据出现困难；但由于计算机技术的迅猛发展，人们生产、搜集、处理数据的能力不断提高。通过使用聚类算法，可将特性相似的数据分为一类，不同特性的数据之间差距很大，为各行各业提供所需要的有用数据，并进行分析。

本节主要介绍了 PAM 算法并将其与 K 均值算法做了比较。K 均值聚类算法是最为经典的，同时也是使用最为广泛的一种基于划分的聚类算法，它属于基于距离的聚类算法。但由于 K 均值算法对孤立点敏感，很可能将孤立点划分到一个类中，使分类的聚类形状扭曲。而 PAM 算法则对孤立点不敏感，因为 PAM 算法会将所有数据进行遍历，以真实的数据作为聚类原型，从而消除了孤立点的影响。

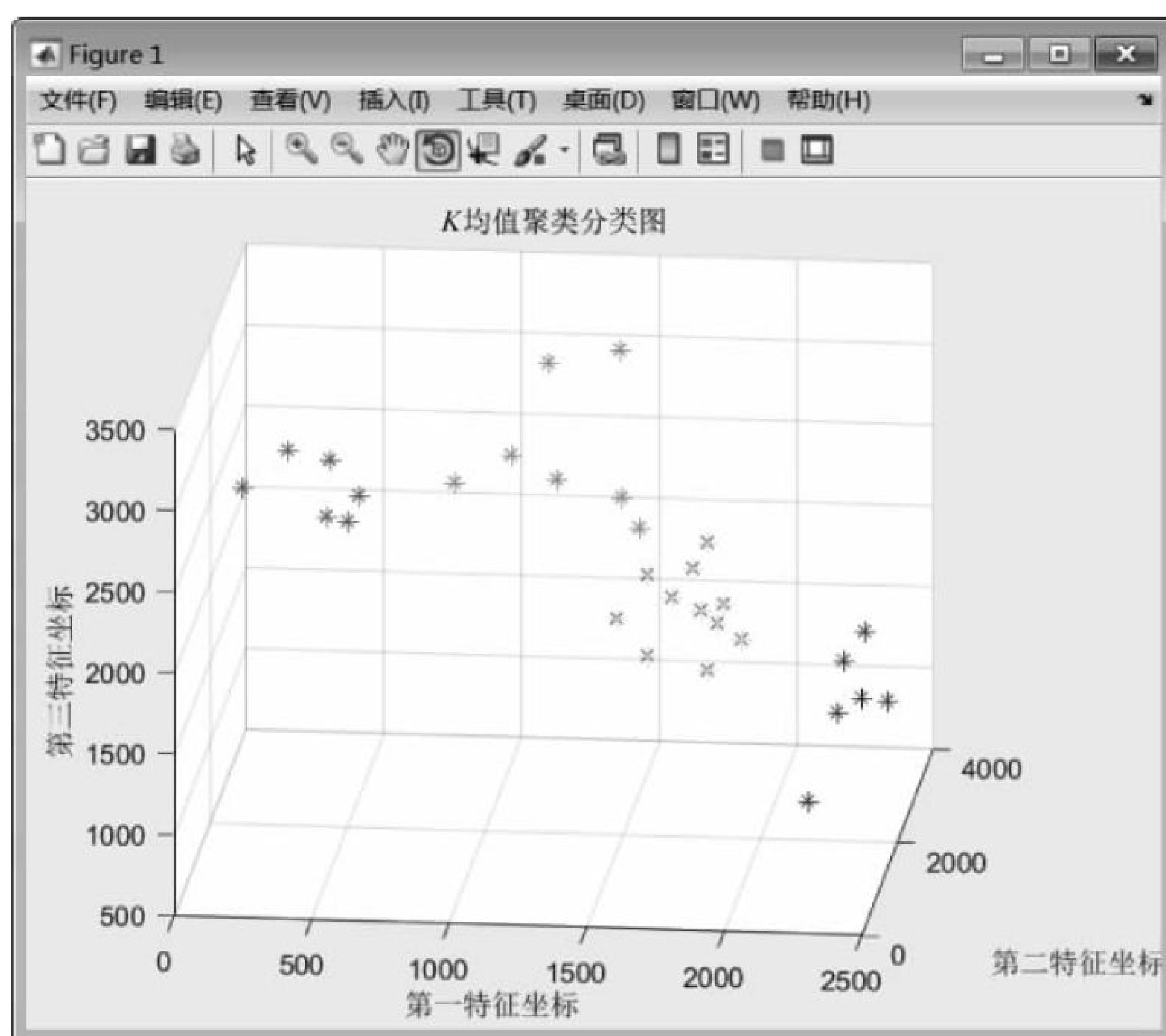


图 4-16 *K* 均值聚类结果图界面

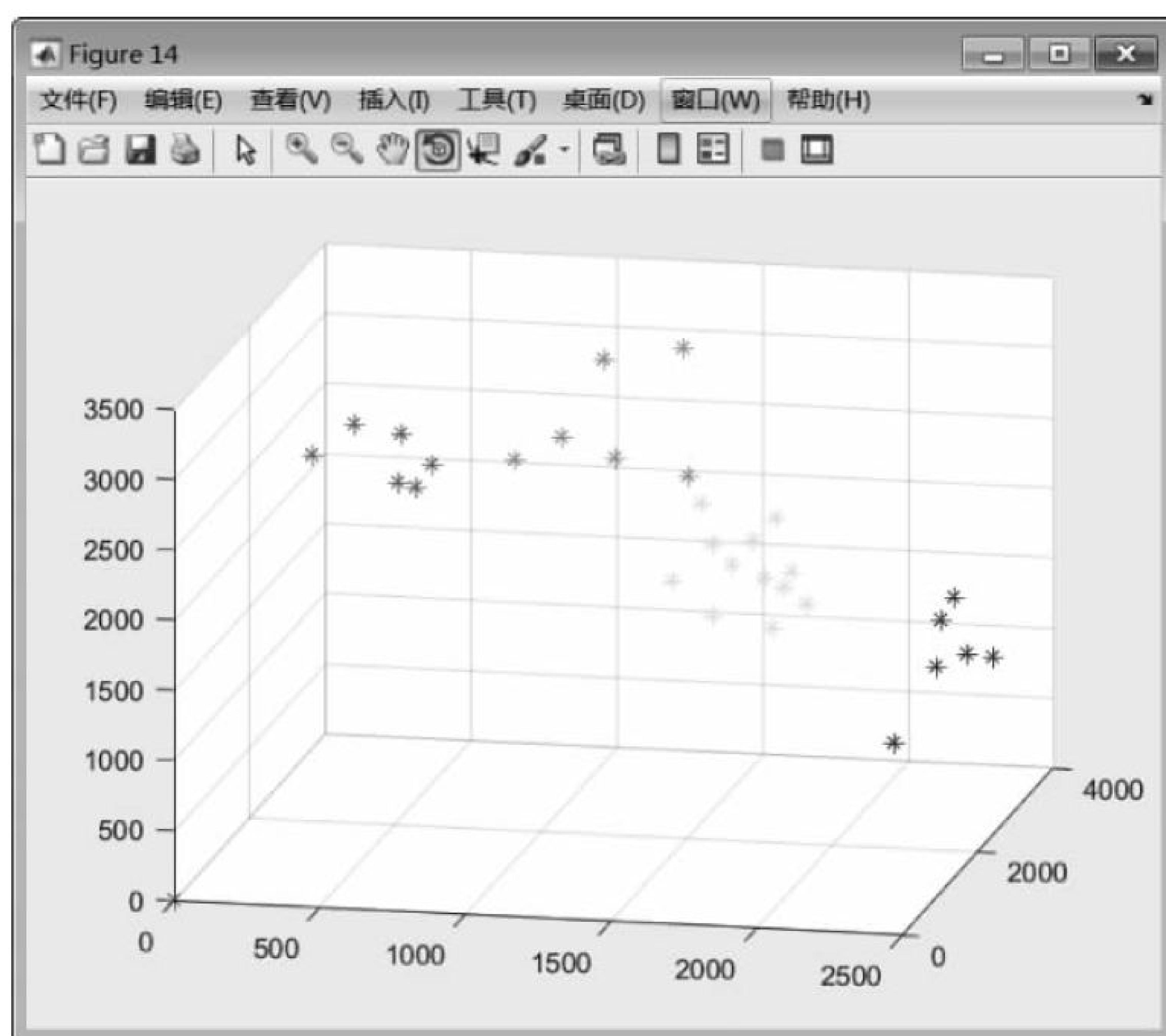


图 4-17 PAM 聚类结果图界面

4.6 数据聚类——层次聚类

4.6.1 层次聚类方法简介

层次聚类就是将数据集按照某种方法进行层次分解,直到满足某种条件为止。按照分类原理的不同,可以分为凝聚和分裂两种方法。层次凝聚的代表是 AGNES 算法,层次分裂的代表是 DIANA 算法。一个完全层次聚类的质量由于无法对已经做的合并或分解进行调整而受到影响。但是层次聚类算法没有使用准则函数,它所含的对数据结构的假设更少,所以它的通用性更强。

4.6.2 凝聚的和分裂的层次聚类

1. 凝聚的层次聚类

凝聚的层次聚类是一种自底向上的策略。典型的方式是,它从每个对象形成自己的簇开始,并且迭代地把簇合并成越来越大的簇,直到所有的对象都在一个簇中,或者满足某个终止条件。该单个簇成为层次结构的根。在合并步骤,它找出两个最接近的簇,并且合并它们,形成一个簇。因为每次迭代合并两个簇,其中每个簇至少包含一个对象,因此凝聚方法最多需要 n 次迭代。

绝大多数层次聚类方法属于这一类,它们只是在簇间相似度的定义上有所不同。凝聚的层次聚类算法过程如图 4-18 所示。

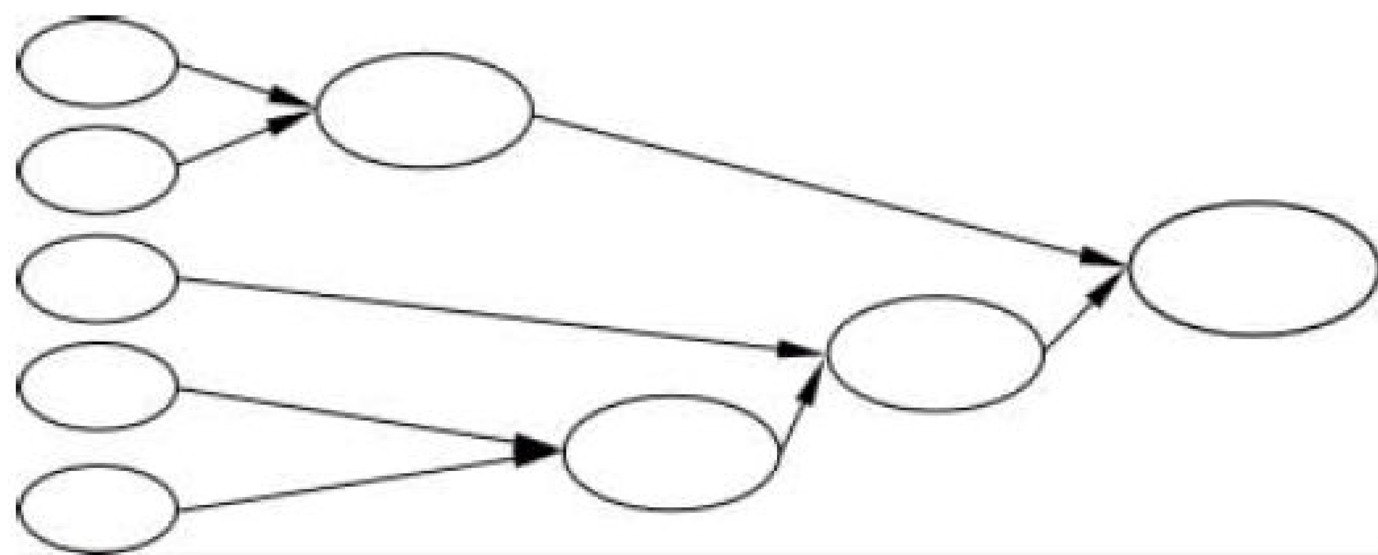


图 4-18 凝聚的层次聚类算法

凝聚方法令每个对象自成一簇,然后这些簇根据某种准则逐步合并。簇合并过程反复进行,直到所有的对象最终合并成一个簇。

2. 分裂的层次聚类

分裂的层次聚类与凝聚的层次聚类相反,采用自顶向下的策略。它从把所有对象置于一个簇中开始,该簇是层次结构的根。然后,它把根上的簇划分成多个较小的簇,并且递归地把这些簇划分成更小的簇,重复划分过程,直到最底层的簇都足够凝聚或者仅包含一个对象,或者簇内的对象都充分相似。分裂的层次聚类算法过程如图 4-19 所示。

所有的对象形成一个初始簇,根据某种原则,将簇分裂,簇的分裂过程反复进行,直到最

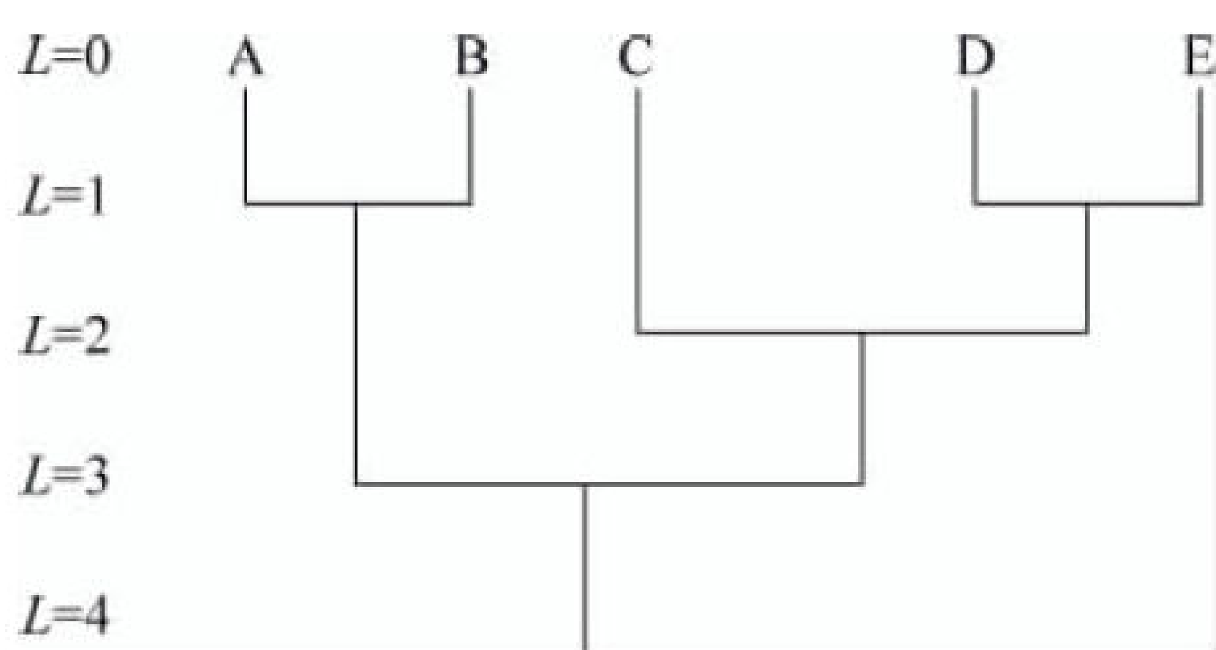


图 4-19 分裂的层次聚类算法

终每个新的簇包含一个对象。

4.6.3 簇间距离度量方法

无论使用凝聚方法还是使用分类方法，一个核心问题是度量两个簇之间的距离，其中每个簇一般是一个对象集。在凝聚和分裂的层次聚类之间，又依据计算簇间的距离的不同，广泛采用 4 种簇间距离度量的方法，其中$|p-p'|$是两个对象或点 p 和 p'之间的距离。

(1) 单连锁(single linkage)又称最近邻(nearest neighbor)方法，指两个不一样的簇之间任意两点之间的最近距离。这里的距离是表示两点之间的相异度，所以距离越近，两个簇相似度越大。这种方法最善于处理非椭圆结构，却对于噪声和孤立点特别的敏感。取出距离很远的两个类之中出现一个孤立点时，这个点就很有可能把两类合并在一起。该方法的距离公式为

$$d_{\min}(c_i,c_j) = \min_{p\in c_i,p'\in c_j}|p-p'| \tag{4-9}$$

(2) 全连锁(complete linkage)又称最远邻(furthest neighbor)方法，指两个不一样的簇中任意两点之间的最远距离。它对噪声和孤立点很不敏感，趋向于寻求某些紧凑的分类，但是，有可能使比较大的簇破裂。该方法的距离公式为

$$d_{\max}(c_i,c_j) = \max_{p\in c_i,p'\in c_j}|p-p'| \tag{4-10}$$

(3) 组平均方法(group average linkage)，定义距离为数据两两距离的平均值。这种方法倾向于合并差异小的两个类，产生的聚类具有相对的健壮性。该方法的距离公式如式(4-11)所示。

$$d_{\text{avg}}(c_i,c_j) = \sum_{p\in c_i}\sum_{p'\in c_j}|p-p'|/n_in_j \tag{4-11}$$

(4) 平均值方法(centroid linkage)，先计算各个类的平均值，然后定义平均值之差为两类的距离。该方法的距离公式为

$$d_{\text{mean}}(c_i,c_j) = |m_i-m_j| \tag{4-12}$$

其中，c_i、c_j 是两个类；m_i、m_j 分别为类 c_i、c_j 的平均值。

当算法使用最小距离来衡量簇间距离时，有时称它为最近邻聚类算法。此外，如果当最近的两个簇之间的距离超过用户给定的阈值时聚类过程就会终止，则称其为单连锁算法。

当一个算法使用最大距离来度量簇间距离时，有时称为最远邻聚类算法。如果当最近两个簇之间的最大距离超过用户给定的阈值时聚类过程便终止，则称其为全连锁算法。

以上最小和最大距离代表了簇间距离度量的两个极端。它们趋向离群点或噪声数据过

分敏感。使用均值距离或平均距离是对最小和最大距离的一种折中方法,并且可以克服离群点敏感性问题。

4.6.4 层次聚类方法存在的不足

在凝聚的层次聚类方法和分裂的层次聚类的所有方法中,都需要用户提供希望得到的聚类的单个数量和阈值作为聚类分析的终止条件,但是对于复杂的数据来说这是很难事先判定的。尽管层次聚类的方法实现起来很简单,但是偶尔也会遇到合并或分裂点抉择困难的情况。这样的抉择是特别关键的,因为只要其中的两个对象被合并或者分裂,接下来的处理将只能在新生成的簇中完成,已形成的处理不能被撤销,两个聚类之间也不能交换对象。如果在某个阶段没有选择合并或分裂的决策,就很可能会导致低质量的聚类结果。此处,这种聚类方法不具备较好的可伸缩性,因为它们合并或分裂的决策需要经过检测和估算大量的对象或簇。

层次聚类算法由于要使用距离矩阵,所以它的时间和空间复杂度都很高 $O(n^2)$,几乎不能在大数据集上使用。层次聚类算法只处理符合某静态模型的簇而忽略了不同簇间的信息,并且忽略了簇间的互连性(指簇间距离较近数据对的多少)和近似度(指簇间数据对的相似度)。

4.6.5 层次聚类的 MATLAB 实现

这里使用平均值方法(centroid linkage)来实现酒瓶颜色的聚类,数据采用表 1-2。下面具体介绍程序算法。

1. 重要代码介绍

(1) 数据标准化处理,程序代码如下:

```
%将数据进行标准化变换
Q1 = zscore(X1);
```

(2) 利用欧拉距离函数计算样本间距离,程序代码如下:

```
%计算样本间的距离
Y1 = pdist(Q1,'euclid') %欧拉距离
```

(3) 用两类间距离定义为类重心间的距离,程序代码如下:

```
Z1 = linkage(Y1,'centroid')
```

2. MATLAB 完整程序

层次聚类的 MATLAB 完整程序代码如下:

```
%层次聚类
clear
X1 = [1739.94    1675.15    2395.96
    373.3      3087.05    2429.47
    1756.77    1652       1514.98
    864.45     1647.31    2665.9
    222.85     3059.54    2002.33
    877.88     2031.66    3071.18
    1803.58    1583.12    2163.05
    2352.12    2557.04    1411.53
    401.3      3259.94    2150.98
    363.34     3477.95    2462.86
    1571.17    1731.04    1735.33
    104.8      3389.83    2421.83
    499.85     3305.75    2196.22
    2297.28    3340.14    535.62
    2092.62    3177.21    584.32
    1418.79    1775.89    2772.9
    1845.59    1918.81    2226.49
    2205.36    3243.74    1202.69
    2949.16    3244.44    662.42
    1692.62    1867.5     2108.97
    1680.67    1575.78    1725.1
    2802.88    3017.11    1984.98
    172.78     3084.49    2328.65
    2063.54    3199.76    1257.21
    1449.58    1641.58    3405.12
    1651.52    1713.28    1570.38
    341.59     3076.62    2438.63
    291.02     3095.68    2088.95
    237.63     3077.78    2251.96];
%数据转化
%将数据进行标准化变换
Q1 = zscore(X1);
%计算样本间的距离
Y1 = pdist(Q1,'euclid') %欧拉距离
D = squareform(Y1)
Z1 = linkage(Y1,'centroid'); %用两类间距离定义为类重心间的距离
T1 = cluster(Z1,4)
%形成聚类
plot3(X1(:,1),X1(:,2),X1(:,3),'*','MarkerSize',8);
grid;
%变颜色
hold on;
for t = 1:length(T1)
    if(T1(t) == 1)
        plot3(X1(t,1),X1(t,2),X1(t,3),'Marker','*','Color','r');
    elseif(T1(t) == 2)
        plot3(X1(t,1),X1(t,2),X1(t,3),'Marker','*','Color','b');
    elseif(T1(t) == 3)
        plot3(X1(t,1),X1(t,2),X1(t,3),'Marker','*','Color','g');
```

```
    elseif(T1(t) == 4)
        plot3(X1(t,1),X1(t,2),X1(t,3),'Marker','*','Color','y');
    end
end
hold on;
xlabel ('X');
ylabel ('Y');
zlabel ('Z');
title('训练数据');
xlabel('样本');
ylabel('类间距离');
title('训练数据');
X2 = [1702.8    1639.79    2068.74
    1877.93    1860.96    1975.3
    867.81     2334.68    2535.1
    1831.49    1713.11    1604.68
    460.69     3274.77    2172.99
    2374.98    3346.98    975.31
    2271.89    3482.97    946.7
    1783.64    1597.99    2261.31
    198.83     3250.45    2445.08
    1494.63    2072.59    2550.51
    1597.03    1921.52    2126.76
    1598.93    1921.08    1623.33
    1243.13    1814.07    3441.07
    2336.31    2640.26    1599.63
    354        3300.12    2373.61
    2144.47    2501.62    591.51
    426.31     3105.29    2057.8
    1507.13    1556.89    1954.51
    343.07     3271.72    2036.94
    2201.94    3196.22    935.53
    2232.43    3077.87    1298.87
    1580.1     1752.07    2463.04
    1962.4     1594.97    1835.95
    1495.18    1957.44    3498.02
    1125.17    1594.39    2937.73
    24.22      3447.31    2145.01
    1269.07    1910.72    2701.97
    1802.07    1725.81    1966.35
    1817.36    1927.4     2328.79
    1860.45    1782.88    1875.13];
figure;
%数据转换
%将数据进行标准化变换
Q2 = zscore(X2);
%计算样本间的距离
Y2 = pdist(Q2,'euclid') %欧拉距离
D1 = squareform(Y2)
Z2 = linkage(Y2,'centroid');
```

```
T2 = cluster(Z2,4)
%形成聚类
plot3(X2(:,1),X2(:,2),X2(:,3),'*','MarkerSize',8);
grid;
%变颜色
hold on;
for t = 1:length(T2)
    if(T2(t) == 1)
        plot3(X2(t,1),X2(t,2),X2(t,3),'Marker','*','Color','r');
    elseif(T2(t) == 2)
        plot3(X2(t,1),X2(t,2),X2(t,3),'Marker','*','Color','b');
    elseif(T2(t) == 3)
        plot3(X2(t,1),X2(t,2),X2(t,3),'Marker','*','Color','g');
    elseif(T2(t) == 4)
        plot3(X2(t,1),X2(t,2),X2(t,3),'Marker','*','Color','y');
    end
end
hold on;
xlabel ('X');
ylabel ('Y');
zlabel ('Z');
title('测试数据');
xlabel('样本 1');
ylabel('类间距离 1');
title('测试数据');
box on
```

运行程序,出现如图 4-20 和图 4-21 所示的聚类结果图界面。

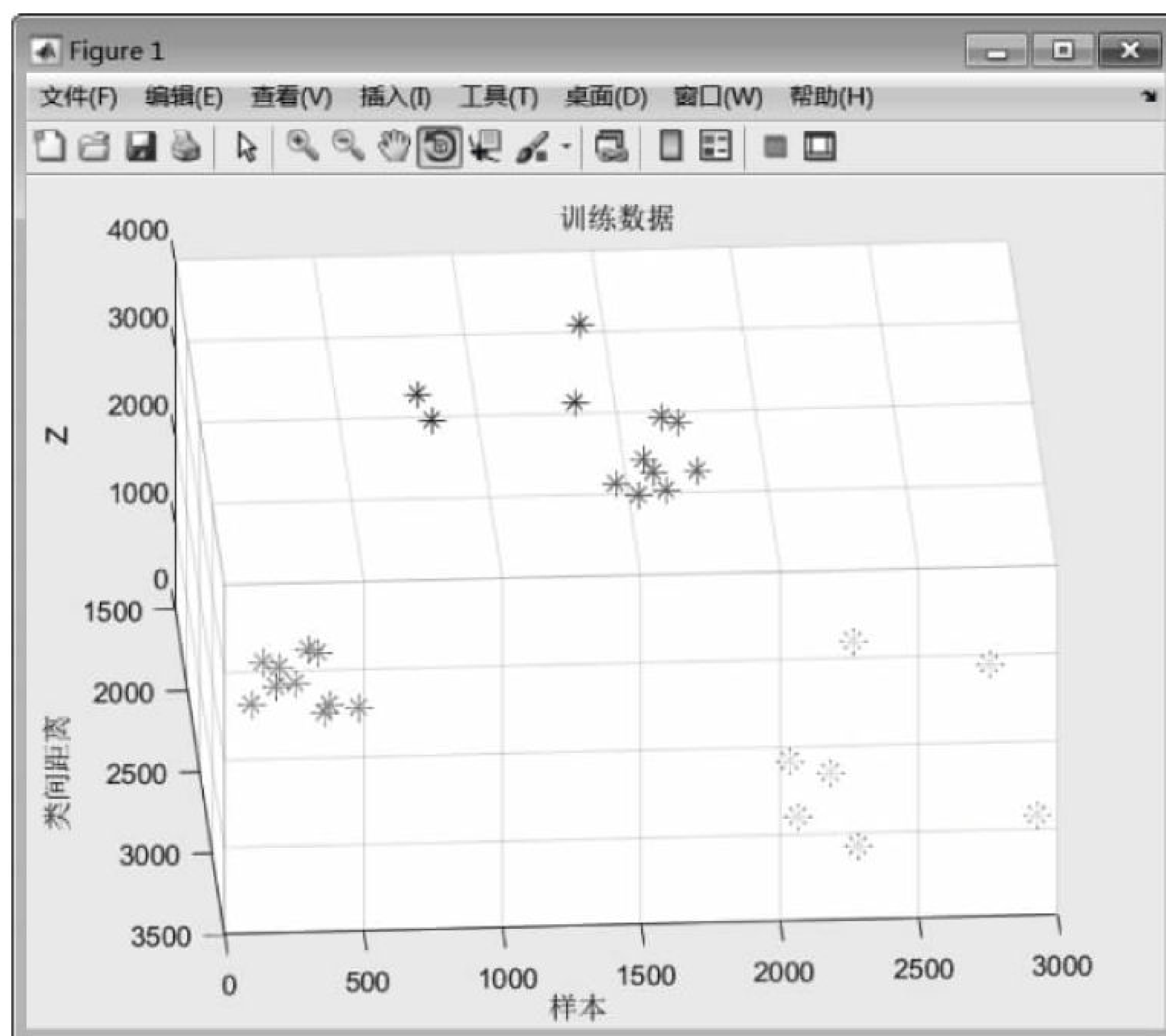


图 4-20 训练数据聚类图界面

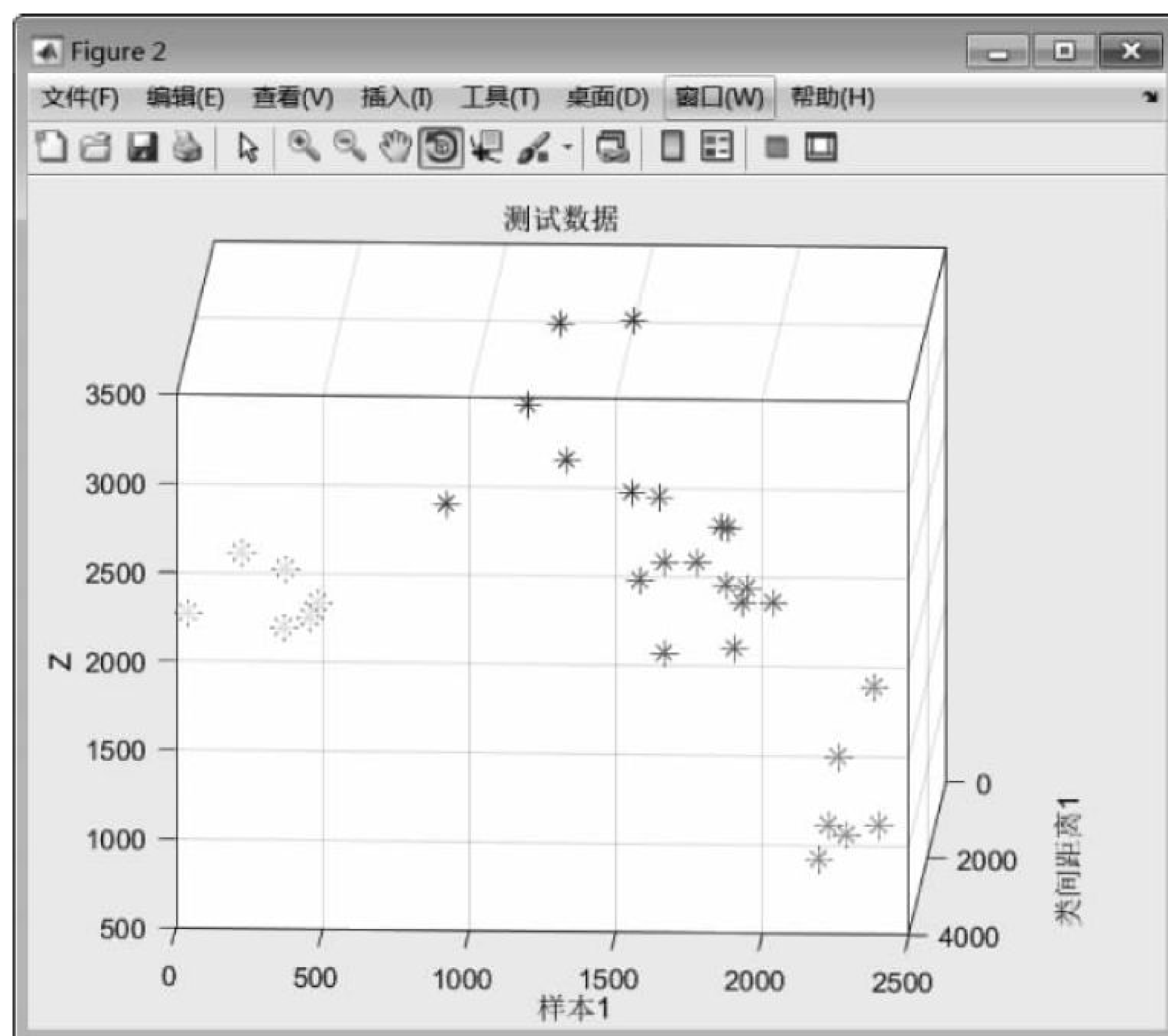


图 4-21 测试数据聚类图界面

程序运行完之后，在命令窗口出现如下运行结果：

```
T1 =
      1
      3
      1
      2
      3
      2
      1
      4
      3
      3
      1
      3
      3
      4
      4
      2
      1
      4
      4
      1
      1
      4
      3
```

```
      4
      2
      1
      3
      3
      3
T2 =
      1
      1
      2
      1
      4
      3
      3
      1
      4
      2
      1
      1
      2
      3
      4
      3
      4
      1
      4
      3
      3
      1
      1
      2
      2
      4
      2
      1
      1
      1
```

从结果中可以看出聚类结果完全正确。

4.6.6 结论

层次聚类方法是聚类分析中应用很广泛的一种方法。它是根据给定的簇间距离度量为准则，构造和维护一棵由簇和子簇形成的聚类树，直至满足某个终结条件为止。其中簇间距离度量方法有最小距离、最大距离、平均值距离和平均距离四种。层次聚类算法简单，而且能够有效处理大数据集，但是它一旦把一组对象合并或者分裂，则已做的处理便不能撤销和

更改。如果某一步没有很好地做好合并或分裂抉择,则可能会导致低质量的聚类效果。

(1) 最大和最小度量代表了簇间距度量的两个极端。它们趋向于对离群点和噪声数据过分敏感。

(2) 使用均值距离和平均距离是对最大和最小距离的一种折中方法,并且可以克服对离群点敏感的问题。

(3) 层次聚类方法尽管简单,但经常会遇到合并或分裂点选择困难的问题。一旦一组对象合并或分裂,下一步的处理将对新生成的簇进行。

(4) 不具有很好的可伸缩性,因为合并或分裂的决定需要检查和估算大量的对象或簇。

(5) 类间距离的定义方法不同,会使分类结果不太一致。实际问题中常用几种不同的方法进行计算,比较其分类结果,从中选择一种比较切合实际的分类方法。

4.7 数据聚类——ISODATA 算法概述

4.7.1 ISODATA 算法应用背景

ISODATA 算法是一种聚类划分算法,称为迭代自组织数据分析或动态聚类。与传统分类方法的根本区别是,它是一种软性分类,而传统聚类划分是硬性划分。在人们不完全了解客观存在的根本属性的情况下,如果使用传统聚类中的非此即彼的思想,最终的结果是比较简单地将分类对象强行划分归类,从而造成对认识过程的伤害。而软性分类可以认识到大多数分类对象在初始认知或初始分类时不太可能显示的最本质属性,这种模糊聚类的过程以一种逐步进化的方式来逼近事物的本质,可以客观地反映人们认识事物的过程,是一种更科学的聚类方式。ISODATA 算法是在没有先验知识的情况下进行分类的,是一种非监督分类的方法,与 K 均值算法有相似之处,即聚类中心通过对均值的迭代运算来决定。但 ISODATA 算法能够吸取中间结果所得的经验,具有自组织性。它通过预先设定的迭代参数,加入了一些试探步骤,并且可以结合成为人机交互的结构,使其能利用中间结果所取得的经验更好地进行分类。

动态聚类的特点在于聚类过程通过不断地迭代来完成,且在迭代中通常允许样本从一个聚合类中转移到另一个聚类中。ISODATA 聚类法认为同类事物在某种属性空间上具有一种密集型的特点,它假定样本集中的全体样本分为 m 类,并选定 Z_K 为初始聚类中心,然后根据最小距离原则将每个样本分配到某一类中;之后不断迭代,计算各类的聚类中心,并以新的聚类中心调整聚类情况,并在迭代过程中,根据聚类情况自动地进行类的合并和分裂。

ISODATA 算法的特点如下:

(1) 无先验知识,启发性推理。

(2) 无监督分类。

ISODATA 算法的基本思想是在每轮迭代过程中,样本重新调整类别之后计算类内及类间有关参数,并和设定的门限比较,确定是两类合并为一类还是一类分裂为两类;不断地“自组织”,以达到在各参数满足设计要求条件下,使各模式到其类心的距离平方和最小。与

K 均值聚类算法相比较，它在下列几方面做了改进：

（1）考虑了类别的合并与分裂，因而有了自我调整类别数的能力。合并主要发生在某一类内样本个数太少的情况，或两类聚类中心之间距离太小的情况。

为此设有最小类内样本数限制，以及类间中心距离参数。若出现两类聚类中心距离小于的情况，可考虑将此两类合并。

分裂主要发生在某一类别的某分量出现类内方差过大的情况下，因而宜分裂成两个类别，以维持合理的类内方差。需要给出一个对类内分量方差的限制参数，用以决定是否需要将某一类分裂成两类。

（2）由于算法有自我调整的能力，因而需要设置若干个控制用参数，如聚类数期望值 K、每次迭代允许合并的最大聚类对数 L，及允许迭代次数 I 等。

下面介绍 ISODATA 算法的步骤。

首先明确几个参数：

N_C：预选聚类中心个数；

K：希望的聚类中心的个数；

θ_N：每个聚类中心的最少样本数；

θ_S：一个聚类域中样本距离分布的样本差；

θ_C：两个聚类中心之间的最小距离；

L：在一次迭代中允许合并的聚类中心的最大对数；

I：允许迭代的次数。

ISODATA 聚类法的详细步骤如下：设有 N 个模式样本 $X_1, X_2, \cdots, X_N$。

第一步：预选 N_C个聚类中心$\{Z_1, Z_2, \cdots, Z_{N_C}\}$，$N_C$不要求等于希望的聚类数目。

第二步：计算每个样本与聚合中心距离，把 N 个样本按最近邻原则（最小距离则）分配到 N_C个聚类中，若 $\| X-Z_j \| = \min\{ \| X-Z_i \|, i=1,2,\cdots,N_C\}$，则 $X \in S_j$。

第三步：判断 S_j 中的样本个数，若 $N_j < \theta_N$，则删除该类，并且 N_C 减去 1，并转至第二步。

第四步：计算分类后的参数，包括各聚类样本中心、类内平均距离及总体平均距离。

各聚类样本中心

$$\boldsymbol{Z}_j = \frac{1}{N_j} \sum_{X \in S_j} X, \quad j = 1,2,\cdots,N_C \tag{4-13}$$

类内平均距离

$$\overline{D}_j = \frac{1}{N_j} \sum_{X \in S_j} \| X - Z_j \|, \quad j = 1,2,\cdots,N_C \tag{4-14}$$

总体平均距离

$$\overline{D} = \frac{1}{N} \sum_{j=1}^{N_C} \sum_{X \in S_j} \| X - Z_j \| = \frac{1}{N} \sum_{j=1}^{N_C} N_j \overline{D}_j \tag{4-15}$$

第五步：根据迭代次数和 N_C 的大小判决算法是分裂，合并还是结束。

（1）如若迭代次数已达到 I 次，即最后一次迭代，则置 $\theta_C = 0$，并且跳到最后一步。

（2）如若 $N_C \leqslant \frac{K}{2}$，即聚类中心数目不大于希望数目的一半，则进入分裂步骤。

(3) 如若 $N_C \geqslant 2K$,即聚类中心数目不小于希望数目的两倍,或者迭代次数为偶数,则进入合并步骤,否则进入分裂步骤。

第六步:分裂步骤如下。

(1) 计算各类类内距离的标准差向量

$$\boldsymbol{\sigma}_j = [\sigma_{j1}, \sigma_{j2}, \cdots, \sigma_{jn}]^{\mathrm{T}}, \quad j = 1, 2, \cdots, N_C \tag{4-16}$$

每一个分量为

$$\boldsymbol{\sigma}_{ij} = \sqrt{\frac{1}{N_j} \sum_{x_{ji} \in X_j} (x_{ji} - z_{ji})^2} \tag{4-17}$$

式中,$i=1,2,\cdots,n$ 是维数;x_{ji} 是 S_j 类的样本 X 的第 i 个分量;z_{ji} 是 S_j 类的聚类中心 Z_j 的第 i 个分量。

(2) 求每个标准差的最大分量,即为 $\sigma_{j\max}$。

(3) 在集合 $\{\sigma_{j\max}\}$ 中,若有 $\sigma_{j\max} > \theta_S$ 说明 S_j 类样本在对应方向上的标准差大于允许值,若同时满足下面两个条件之一:

① $\overline{D}_j > \overline{D}$ 和 $N_j > 2(\theta_N + 1)$

② $N_C \leqslant \frac{K}{2}$

则 Z_j 分裂成 Z_j^+ 和 Z_j^-,N_C 加 1。Z_j^+ 构成为:Z_j 中对应 $\sigma_{j\max}$ 分量加上 $k\sigma_{j\max}$;Z_j^- 构成为:Z_j 中对应 $\sigma_{j\max}$ 分量减去 $k\sigma_{j\max}$,$0<k<1$,k 为分裂系数。若完成分裂,迭代次数加一,转回第二步,否则继续下一步。

第七步:合并步骤如下。

(1) 计算所有聚类中心之间的距离

$$D_{ij} = \| Z_i - Z_j \|, \quad \begin{aligned} & i = 1, 2, \cdots, N_C - 1; \\ & j = i+1, i+2, \cdots, N_C \end{aligned} \tag{4-18}$$

(2) 比较所有的 D_{ij} 与 θ_C 的值,将小于 θ_C 的 D_{ij} 按升序排列,形成集合 $\{D_{i_1,j_1}, D_{i_2,j_2}, \cdots, D_{i_L,j_L}\}$。

(3) 将集合 $\{D_{i_1j_1}, D_{i_2j_2}, \cdots, D_{i_Lj_L}\}$ 中每个元素对应的两类合并,得到新的聚类,其中心为

$$\boldsymbol{Z}_l^* = \frac{1}{N_{i_l} + N_{j_l}}(N_{i_l}\boldsymbol{Z}_{i_l} + N_{j_l}\boldsymbol{Z}_{j_l}), \quad l = 1, 2, \cdots, L \tag{4-19}$$

每合并一对,N_C 减 1。

第八步:如若是最后一次迭代运算,则算法结束;否则有两种情况,需要操作者修改参数时,跳到第一步,输入参数不需要改变时,跳到第二步,选择两者之一,迭代次数加 1,然后继续进行运算。

4.7.2 ISODATA 算法的 MATLAB 实现

这里还采用表 1-2 所示数据。

运用 MATLAB 语言来实现 ISODATA 算法的主要思想是,把类的分裂、合并操作看成是一种三维数组中行向量位置移动的过程,每一个样本作为数组中的一个行向量,而每一行

的每一列都是样本的属性值；使用 MATLAB 的矩阵运算就可以完成对样本位置的调整，从而模拟对类的调整，最终达到聚类分析的结果。程序实现流程：将输入样本初始化分类之后，对分类进行判断，是分裂、合并还是终止，然后再根据判断结果进行分类，如此周而复始。

1. ISODATA 算法的重要源代码

(1) 初始化程序代码如下：

```
%step1:初始化
T = input('是否要设置输入参数?是请输入"1",否请输入"0":T=');
if(T==1)
K = input('请输入预期聚类中心数目:K=');                    %预期的聚类中心个数
Qn = input('请输入每一聚类中最少样本数:Qn=');               %每一类中最少的样本数目
Qs = input('请输入一个聚类中样本距离分布的标准差:Qs=');%一个聚类中样本距离分布的标准差
Qc = input('请输入两类聚类中心间的最小距离:Qc=');          %两类聚类中心间的最小距离
%L = input('请输入一次迭代中可以合并聚类中心的最多个数:L=');%一次迭代中可以合并聚类
                                                          %中心的最多个数
end
% K = 3;                                                    %预期的聚类中心个数
% Qn = 5;                                                   %每一类中最少的样本数目
% Qs = 1.8;                                                 %一个聚类中样本距离分布的标准差
% Qc = 1.5;                                                 %两类聚类中间的最小距离
seperate = 1;                         %分裂标识,为1时可进入分裂循环,为0时跳出分裂循环
while(seperate==1)
% disp('正在运行 while 循环');
```

这里参数的设置如下：

```
是否要设置输入参数?是请输入"1",否请输入"0":T=1
请输入预期聚类中心数目:K=4
请输入每一聚类中最少样本数:Qn=2
请输入一个聚类中样本距离分布的标准差:Qs=1
请输入两类聚类中心间的最小距离:Qc=4
```

(2) 将待分类数据分别分配给距离最近的聚类中心程序，代码如下：

```
%step2:将待分类数据分别分配给距离最近的聚类中心
distance = zeros(n,Nc);%n 为 x 的行数,NC 为初始聚类中心个数
for i=1:n
for j=1:Nc
distance(i,j) = norm(x(i,:)-center(j,:));      %遍历到聚类中心的欧氏距离
end
end
[m,index] = min(distance,[],2);                 %逻辑索引 index 为 1 或 0;
class = index;
clear m;
clear index;
```

```
clear distance;                          % 删除 m,index,distance;
%统计各子集的样本数目
num = zeros(1,Nc);
for i = 1:Nc
index = find(class == i);                % 找到第 i 行的索引和值;
num(i) = length(index);                  % 子集 i 的样本数目
end
clear i;
clear index;
```

(3) 取消样本数目小于 Qn 的子集程序,代码如下:

```
%step3:取消样本数目小于 Qn 的子集
index = find(num >= Qn);                 % Qn 为每一类中最少样本数目;
Nc = length(index);
center_hat = zeros(Nc,d);
for i = 1:Nc
center_hat(i,:) = center(index(i),:);
end
center = center_hat;
clear center_hat;
clear index;
%重新将待分类数据分别分配给距离最近的聚类中心
distance = zeros(n,Nc);
for i = 1:n
for j = 1:Nc
distance(i,j) = norm(x(i,:) - center(j,:));
end
end
[m,index] = min(distance,[],2);
class = index;
clear m;
clear index;
clear distance;
```

(4) 计算分类后的参数的程序,即各聚类样本中心、类内平均距离及总体平均距离,代码如下:

```
%step4:修正聚类中心
new_center = zeros(Nc,d);
num = zeros(1,Nc);
for i = 1:Nc
index = find(class == i);
num(i) = length(index);                  % 子集 i 的样本数目
new_center(i,:) = mean(x(index,:));      % 子集 i 的聚类中心
end
center = new_center;
clear new_center;
clear index;
```

```
%step5:计算各子集中的样本到中心的平均距离 dis
%step6:计算全部模式样本与其对应聚类中心总平均距离 ddis
dis = zeros(1,Nc);
ddis = 0;
for i=1:Nc
index = find(class==i);
for j=1:num(i)
dis(i) = dis(i)+norm(x(index(j),:)-center(i,:));
end
ddis = ddis+dis(i);
dis(i) = dis(i)/num(i);
end
ddis = ddis/n;
clear index;
```

（5）判断分裂、合并及迭代的程序，代码如下：

```
%step7:判断分裂、合并及迭代
%如果迭代次数达到 Imax 次,置 Qc=0,跳出循环至 step14
if I==Imax %(1)
Qc = 0;
break;
end
if (Nc<=K/2) %(2)如果不进入分裂则跳到 step11,合并
seperate = 1;
end
if(mod(I,2)==0|Nc>=2*K) %(3)
break;
else
seperate = 1;
end
```

（6）分裂的相关程序代码如下：

```
%step8:分裂
%计算每个聚类中,各样本到中心的标准差向量
sigma = zeros(Nc,d);     %sigma(i)代表第 i 个聚类的标准差向量
for i=1:Nc
index = find(class==i);
for j=1:num(i)
sigma(i,:) = sigma(i,:)+(x(index(j),:)-center(i,:)).^2;
end
sigma(i,:) = sqrt(sigma(i,:)/num(i));
end
clear index;
%step9:求各个标准差{sigma_j}的最大分量
[sigma_max,max_index] = max(sigma,[],2);
%step10:分裂
k=0.5;                    %分裂聚类中心时使用的系数
```

```
temp_Nc = Nc;
for i = 1:temp_Nc
if sigma_max(i)> Qs&((dis(i)> ddis&num(i)> 2 * (Qn + 1))|Nc <= K/2)
Nc = Nc + 1;
 % 将 z(i)分裂为两个新的聚类中心
center(Nc,:) = center(i,:);
center(i,max_index(i)) = center(i,max_index(i)) + k * sigma_max(i);
center(Nc,max_index(i)) = center(Nc,max_index(i)) + k * sigma_max(i);
end
end
record(I) = Nc;
```

(7) 合并的相关程序,代码如下:

```
 %% step11:合并
   %计算全部聚类中心间的距离
center_Dis = zeros(Nc - 1,Nc);
for i = 1:Nc
for j = i + 1:Nc
center_Dis(i,j) = norm(center(i,:) - center(j,:));
end
end
 %% step12,13:如果距离最小的两个中心之间距离小于 Qc,将其合并
 % 找出距离最近的两个中心
min_Dis = center_Dis(1,3);      % 最小距离
min_index = [1,3];              % 距离最近的两个中心的标号
for i = 1:Nc
for j = i + 1:Nc
if center_Dis(i,j)< min_Dis
min_Dis = center_Dis(i,j);
min_index = [i,j];
end
end
end
if min_Dis < Qc
 % 合并距离最近的两个中心
 % 合并产生的新中心为
new_center = (center(min_index(1),:) * num(min_index(1)) + center(min_index(2),:) * num(min_
index(2)))/(num(min_index(1)) + num(min_index(2)));
temp_center = zeros(1,Nc);
temp_center = center;
temp_center(min_index(1),:) = new_center;
temp_center(min_index(2),:) = center(Nc,:);
Nc = Nc - 1; % 聚类数目减 1
center = temp_center(1:Nc,:);
clear temp_center
end
end
```

```
record(I) = Nc;
I = I + 1
%x 为聚类中元素矩阵,z 为聚类中心,k 为聚类元素维数,n 为 x 中元素个数
function [delta] = clusterStd(x,z,n,k)
d = zeros(1,k);
for i = 1:n
    for j = 1:k
        d(j) = d(j) + (x(i,j) - z(i))^2;
    end
end
delta = sqrt(d./n);
```

2. ISODATA 的 MATLAB 完整程序

完整 MATLAB 程序代码如下：

```
close all;
clear all;
clc;
%数据导入
in_data = load('SelfOrganizationSimulation.dat');          %导入数据
x = in_data;                                                %待分类样本
%给数据添加类别标签
label = [ones(10,1);ones(10,1) * 2;ones(10,1) * 3];
in_data = [in_data,label];
% ------------- ISODATA --------------- %
Imax = 6;                                                   %迭代次数
Nc = 5;                                                     %预选初始聚类中心个数
%%记录聚类数目
record = zeros(1,Imax);
%随机选取 Nc 个初始聚类中心
r = randperm(30);
for i = 1:Nc
center(i,:) = x(r(i),:);
end
clear i;
clear r;
[n,d] = size(x);                                            %n 为数据行数,d 为数据列数
I = 1;
while I < Imax
%step1:初始化
T = input('是否要设置输入参数?是请输入"1",否请输入"0": T = ');
if(T == 1)
K = input('请输入预期聚类中心数目: K = ');                  %预期的聚类中心个数
Qn = input('请输入每一聚类中最少样本数: Qn = ');            %每一类中最少的样本数目
Qs = input('请输入一个聚类中样本距离分布的标准差: Qs = '); %一个聚类中样本距离分布的标准差
Qc = input('请输入两类聚类中心间的最小距离: Qc = ');        %两类聚类中心间的最小距离
%L = input('请输入一次迭代中可以合并聚类中心的最多个数: L = '); %一次迭代中可以合并聚类
                                                            %中心的最多个数
```

```
end
% K = 3;                      % 预期的聚类中心个数
% Qn = 5;                     % 每一类中最少的样本数目
% Qs = 1.8;                   % 一个聚类中样本距离分布的标准差
% Qc = 1.5;                   % 两类聚类中间的最小距离
seperate = 1;                 % 分裂标识,为 1 时可进入分裂循环,为 0 时跳出分裂循环
while(seperate == 1)
% disp('正在运行 while 循环');
% step2:将待分类数据分别分配给距离最近的聚类中心
distance = zeros(n,Nc);       % n 为 x 的行数,Nc 为初始聚类中心个数
for i = 1:n
for j = 1:Nc
distance(i,j) = norm(x(i,:) - center(j,:));   % 遍历到聚类中心的欧氏距离
end
end
[m,index] = min(distance,[],2);               % 逻辑索引 index 为 1 或 0;
class = index;
clear m;
clear index;
clear distance;                               % 删除 m,index,distance;
% 统计各子集的样本数目
num = zeros(1,Nc);
for i = 1:Nc
index = find(class == i);                     % 找到第 i 行的索引和值;
num(i) = length(index);                       % 子集 i 的样本数目
end
clear i;
clear index;
% step3:取消样本数目小于 Qn 的子集
index = find(num >= Qn);                      % Qn 为每一类中最少样本数目;
Nc = length(index);
center_hat = zeros(Nc,d);
for i = 1:Nc
center_hat(i,:) = center(index(i),:);
end
center = center_hat;
clear center_hat;
clear index;
% 重新将待分类数据分别分配给距离最近的聚类中心
distance = zeros(n,Nc);
for i = 1:n
for j = 1:Nc
distance(i,j) = norm(x(i,:) - center(j,:));
end
end
[m,index] = min(distance,[],2);
class = index;
clear m;
clear index;
clear distance;
```

```
%step4:修正聚类中心
new_center = zeros(Nc,d);
num = zeros(1,Nc);
for i=1:Nc
index = find(class==i);
num(i) = length(index);                    %子集 i 的样本数目
new_center(i,:) = mean(x(index,:));        %子集 i 的聚类中心
end
center = new_center;
clear new_center;
clear index;
%step5:计算各子集中的样本到中心的平均距离 dis
%step6:计算全部模式样本与其对应聚类中心总平均距离 ddis
dis = zeros(1,Nc);
ddis = 0;
for i=1:Nc
index = find(class==i);
for j=1:num(i)
dis(i) = dis(i)+norm(x(index(j),:)-center(i,:));
end
ddis = ddis+dis(i);
dis(i) = dis(i)/num(i);
end
ddis = ddis/n;
clear index;
%step7:判断分裂,合并及迭代
%如果迭代次数达到 Imax 次,置 Qc=0,跳出循环至 step14
if I==Imax %(1)
Qc = 0;
break;
end
if (Nc<=K/2) %(2)如果不进入分裂则跳到 step11,合并
seperate = 1;
end
if(mod(I,2)==0|Nc>=2*K) %(3)
break;
else
seperate = 1;
end
%step8:分裂
%计算每个聚类中,各样本到中心的标准差向量
sigma = zeros(Nc,d);                        %sigma(i)代表第 i 个聚类的标准差向量
for i=1:Nc
index = find(class==i);
for j=1:num(i)
sigma(i,:) = sigma(i,:)+(x(index(j),:)-center(i,:)).^2;
end
sigma(i,:) = sqrt(sigma(i,:)/num(i));
end
clear index;
```

```
%step9:求各个标准差{sigma_j}的最大分量
[sigma_max,max_index] = max(sigma,[],2);
%step10:分裂
k = 0.5; %分裂聚类中心时使用的系数
temp_Nc = Nc;
for i = 1:temp_Nc
if sigma_max(i)> Qs&((dis(i)> ddis&num(i)> 2 * (Qn + 1))|Nc <= K/2)
Nc = Nc + 1;
%将 z(i)分裂为两个新的聚类中心
center(Nc,:) = center(i,:);
center(i,max_index(i)) = center(i,max_index(i)) + k * sigma_max(i);
center(Nc,max_index(i)) = center(Nc,max_index(i)) + k * sigma_max(i);
end
end
record(I) = Nc;
%绘制聚类效果图
figure;
for i = 1:30
if class(i) == 1
plot3(x(i,1),x(i,2),x(i,3),'r * ');      %红色 * 表示第 1 簇
hold on;
end
if class(i) == 2
plot3(x(i,1),x(i,2),x(i,3),'b + ');      %蓝色 + 表示第 2 簇
hold on;
end
if class(i) == 3
plot3(x(i,1),x(i,2),x(i,3),'go');        %绿色 o 表示第 3 簇
hold on;
end
if class(i) == 4
plot3(x(i,1),x(i,2),x(i,3),'kx');        %黑色 x 表示第 4 簇
hold on;
end
if class(i) == 5
plot3(x(i,1),x(i,2),x(i,3),'md');        %品红色菱形表示第 5 簇
hold on;
end
end
title('聚类效果图');
I = I + 1;
end
% disp('正在运行合并');
if(I < Imax);
%% step11:合并
  %计算全部聚类中心间的距离
center_Dis = zeros(Nc - 1,Nc);
for i = 1:Nc
for j = i + 1:Nc
center_Dis(i,j) = norm(center(i,:) - center(j,:));
```

```
end
end
%% step12,13:如果距离最小的两个中心之间距离小于 Qc,将其合并
% 找出距离最近的两个中心
min_Dis = center_Dis(1,3);                    % 最小距离
min_index = [1,3];                            % 距离最近的两个中心的标号
for i = 1:Nc
for j = i + 1:Nc
if center_Dis(i,j)< min_Dis
min_Dis = center_Dis(i,j);
min_index = [i,j];
end
end
end
if min_Dis < Qc
% 合并距离最近的两个中心
% 合并产生的新中心为
new_center = (center(min_index(1),:) * num(min_index(1)) + center(min_index(2),:) * num(min_
index(2)))/(num(min_index(1)) + num(min_index(2)));
temp_center = zeros(1,Nc);
temp_center = center;
temp_center(min_index(1),:) = new_center;
temp_center(min_index(2),:) = center(Nc,:);
Nc = Nc - 1; % 聚类数目减 1
center = temp_center(1:Nc,:);
clear temp_center
end
end
record(I) = Nc;
I = I + 1
% 绘制聚类效果图
hold off;
for i = 1:30
if class(i) == 1
plot3(x(i,1),x(i,2),x(i,3),'r * ');           % 红色 * 表示第 1 簇
hold on;
end
if class(i) == 2
plot3(x(i,1),x(i,2),x(i,3),'b + ');           % 蓝色 + 表示第 2 簇
hold on;
end
if class(i) == 3
plot3(x(i,1),x(i,2),x(i,3),'go');             % 绿色 o 表示第 3 簇
hold on;
end
if class(i) == 4
plot3(x(i,1),x(i,2),x(i,3),'kx');             % 黑色 x 表示第 4 簇
hold on;
end
```

```
if class(i) == 5
plot3(x(i,1),x(i,2),x(i,3),'md');                    %品红色菱形表示第5簇
hold on;
end
if class(i) == 6
plot3(x(i,1),x(i,2),x(i,3),'c.');                    %青色实心点表示第6簇
hold on;
end
if class(i) == 7
plot3(x(i,1),x(i,2),x(i,3),'yp');                    %黄色五角星表示第7簇
hold on;
end
end
title('聚类效果图');
end
Nc
center
class;
%%
% figure(2);
% plot(record);
% title('聚类数目变化曲线');
%%统计聚类效果
%%result(i,j)代表第i类数据被聚类至第j簇的数量
% result = zeros(3,Nc);
% clear m;
% for m = 1:3
% for i = (50 * (m - 1) + 1):1:50 * m
% for j = 1:Nc
% if class(i) == j
% result(m,j) = result(m,j) + 1;
% end
% end
% end
% end
%%计算准确率、召回率、F值
%%P(i,j)代表第i类数据与第j簇相应的准确率
%%R(i,j)代表第i类数据与第j簇相应的召回率
% for i = 1:3
% for j = 1:Nc
% P(i,j) = result(i,j)/num(j);
% R(i,j) = result(i,j)/50;
% F(i,j) = 2 * P(i,j) * R(i,j)/(P(i,j) + R(i,j));
%
%% disp('F(i,j)代表第i类数据与第j簇相应的F值');
%% F
% disp('FF(i)代表第i类数据的F值');
% FF = max(F,[],2)
% disp('整个聚类结果的F值')
% F_final = mean(FF);
```

程序运行完之后，得到聚类结果图界面如图 4-22 所示。

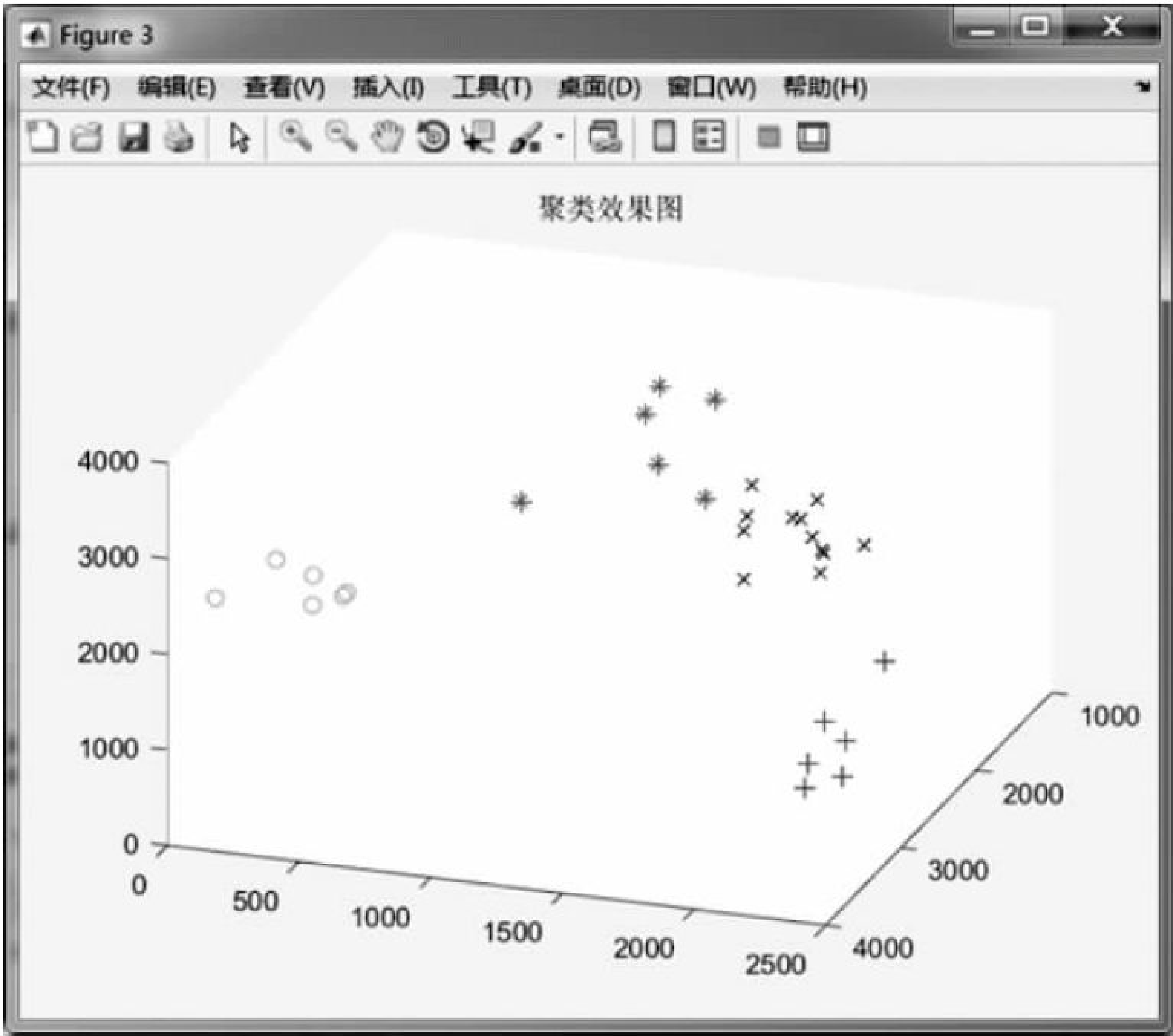


图 4-22 ISODATA 算法聚类结果图界面

程序运行完之后，在命令窗口出现如下结果：

```
是否要设置输入参数?是请输入"1",否请输入"0": T = 1
请输入预期聚类中心数目: K = 4
请输入每一聚类中最少样本数: Qn = 2
请输入一个聚类中样本距离分布的标准差: Qs = 1
请输入两类聚类中心间的最小距离: Qc = 4
I =
      7
Nc =
      4
center =
    1.0e + 03 *
    1.2492     1.9473     2.9441
    0.3012     3.2749     2.2052
    2.2603     3.0410     1.0579
    1.7434     1.7495     2.0070
ans =
1 ～ 27 列
4    4    1    4    2    3    3    4    2    1    4    4    1    3
2    3    2    4    2    3    3    4    4    1    1    2    1
28 ～ 30 列
4    4    4
```

4.7.3 结论

MATLAB编程实现的优点主要是通过模拟ISODATA算法的思想,根据同类样本分布密集性的特点很容易归类。但是,由于大量采用矩阵运算,每次迭代都会产生新的重排矩阵,对于计算时间和空间并不是一种很好的利用。特别是针对ISODATA这种算法,在样本数目非常大时是非常费时的,因此有必要进一步改进。

习题

(1) 什么是聚类? 聚类的准则是什么?

(2) 简述 K 均值聚类的原理。

(3) 简述 K 均值算法的优缺点。

(4) 简述 K 均值算法、KNN算法及PAM算法的区别。

(5) 简述层次聚类算法的原理。

(6) 简述IOSDATA算法的原理。

第5章

模糊聚类分析

一个古老的希腊悖论是:“一粒种子肯定不叫一堆,两粒也不是,三粒也不是……然而,所有的人都同意,一亿粒种子肯定叫一堆。那么,适当的界限在哪里?我们能不能说,123 585粒种子不叫一堆而123 586粒就构成一堆呢?”

这一古老的问题向“精确”求解问题提出了挑战。那么“模糊”是否可以给这一古老的问题画上圆满的句号呢?

5.1 模糊逻辑的发展

许多概念没有一个清晰的外延,比如我们不能在年龄上画线,线内是年轻人,而在线外就是老年人;另外,有些概念本身具有开放性,比如智慧,我们不可能列举出应满足的全部条件。因此,出现了“模糊”。模糊性是伴随着复杂性而出现的,比如判断一个人是否年轻,可能就会从年龄、外貌、心态等方面综合考察。模糊性也是起源于事物的发展变化性的,比如人从年轻逐渐走向年老,这一过程是渐变的,处于过渡阶段的事物的基本特征是不确定的,其类属是不清楚的。所以,总是存在不确定性,即模糊。

有关模糊逻辑的第一次发表要追溯到1965年。美国加利福尼亚大学伯克利分校的系统理论专家L. A. Zadeh教授把经典集合与J. Lukasievicz的多值逻辑融为一体,创立了模糊逻辑理论。

模糊逻辑的首次应用发生在欧洲。1974年,英国伦敦Queen Mary学院的E. H. Mamdani教授使用模糊逻辑控制不能使用传统技术控制的蒸汽机,从而开创了模糊控制的历史。

之后,德国亚琛工业大学的Hans-Jürgen Zimmermann将模糊逻辑用于决策支持系统。随后,模糊逻辑相继应用到其他工业领域,如多变量非线性热水场的控制和水泥窑的控制等,但此时模糊逻辑在工业上仍未得到广泛肯定。而为数不多的使用模糊逻辑的应用也通过使用多值逻辑或连续逻辑限制模糊逻辑,从而掩盖了模糊逻辑的思维模式。

在欧洲,从1980年左右开始,模糊逻辑在决策支持和数据分析应用方面势头强劲。

5.2 模糊集合

医生在评估就诊者是否患有重感冒时，在脑子中没有精确的阈值，那么他们是如何下定论的呢？心理学研究已经表明：医生在做出结论时要与两个“原型”对照，一个“原型”为理想的重感冒患者，症状是脸色苍白、出汗并伴有寒战；另一个“原型”为没有发热且没有发热征兆的健康人。医生参照这两个极端确诊就诊者属于这两个极端的程度。

5.2.1 由经典集合到模糊集合

如何对医生诊断过程建立数学模型？根据集合理论，首先定义一个包括所有重感冒患者的集合，然后定义一个数学函数，用于表明每一个患者是否属于这个集合。在传统数学中，这个指标函数可以唯一鉴定每一个患者是集合的成员或非成员，如图 5-1 所示。图中黑色区域为“患重感冒的患者”的集合，体温高于 102℉的患者属于重感冒患者。

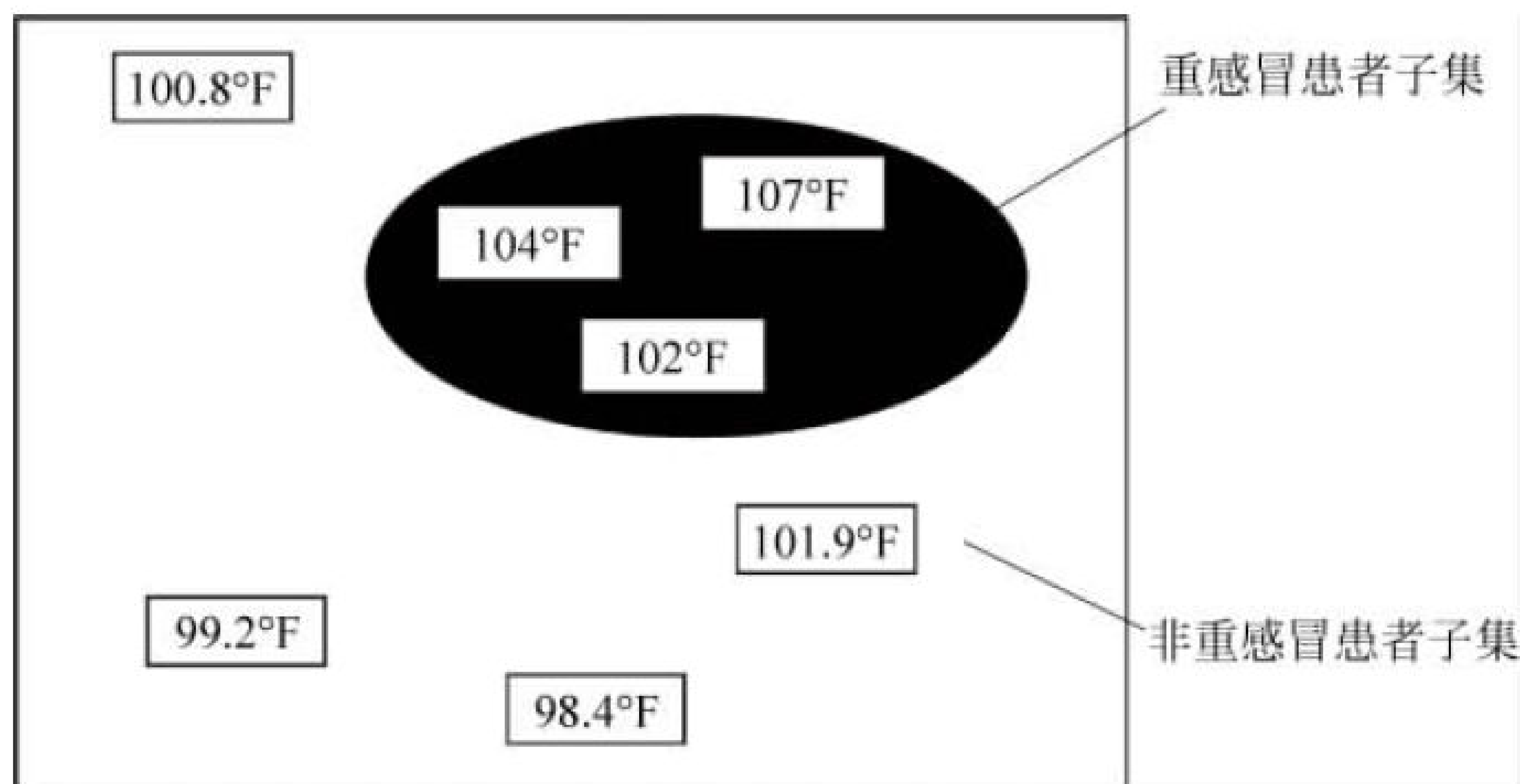


图 5-1　经典集合表示重感冒患者集合(体温高于 102℉的患者属于重感冒患者)

在经典集合中涉及如下概念。

论域：被讨论对象的全体，又称为全域，通常用大写字母 U、E、X、Y 等来表示。

元素：组成某个集合的单个对象称为该集合的一个元素，通常用小写 a、b、x、y 等来表示。

子集：由同一集合的部分元素组成的一个新集合，称为原集合的一个子集，通常用大写字母 A、B、C 等来表示。

通常将集合分为有限集(含有有限多个元素)和无限集(含有无限多个元素)。有限集常用枚举法表示，如 $A=\{x_1,x_2,\cdots,x_n\}$，表明集合 A 含有 n 个元素。如果对象个体 x 是属于 A 的一个元素时，就记为 $x\in A$，读作 x 属于 A；如果对象个体 x 不是集合 A 的元素时，就记为 $x\notin A$，读作 x 不属于 A。无限集常用描述法表示，如 $B=\{x\mid x>2\}$，表明所有大于 2 的数都属于集合 B。

经典集合还有一种表示方法，即特征函数(或隶属度函数)法，它用特征函数来确定一个集合。

设集合 A 是论域 U 的一个子集。所谓 A 的特征函数 $\chi_A(x)$：$\forall x\in U$，若 $x\in A$，则规定 $\chi_A(x)=1$；否则 $\chi_A(x)=0$，即

$$\chi_A(x)=\begin{cases}1, & x\in A\\ 0, & x\notin A\end{cases}$$

任一特征函数都唯一确定了一个集合。也就是说，对于经典集合，论域 U 中的任何一个元素 x，对于某一确定的集合 A，要么 $x\in A$，要么 $x\notin A$。特征函数示意图如图 5-2 所示。

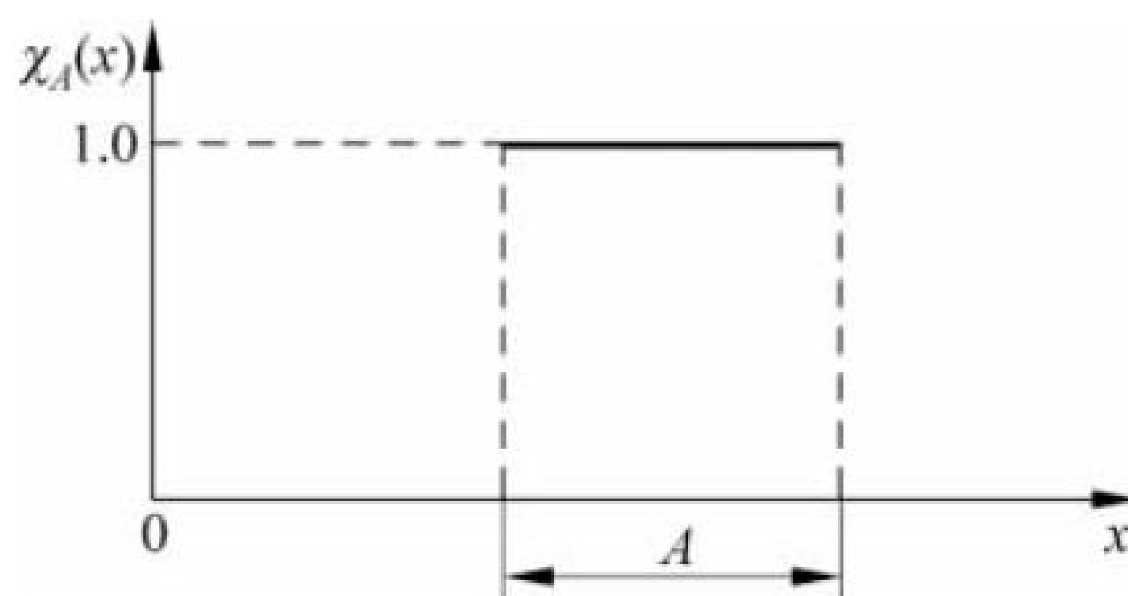


图 5-2 特征函数示意图

显然，如果想根据患者是重感冒患者还是非重感冒患者来定义一个 U 上的经典集合，将存在一定的困难。对于某些不具有清晰边界的集合，经典集合无法定义。

由于经典理论存在这样的局限性，而人们又希望使用集合的概念表述模糊的事物，这就需要新的理论弥补经典集合的局限性，因而引出了模糊集合理论。

5.2.2 模糊集合的基本概念

模糊集合论是一门用清晰的数学方法描述边界不清的事物的数学理论。1965 年美国教授 L. A. Zadeh 将经典集合里的特征函数的取值范围由{0,1}扩展到闭区间[0,1]，认为某一事物属于某个集合的特征函数不仅只有 0 或 1，而是可以取 0～1 的任何数值，即一个事物属于某个集合的程度，可以是 0～1 的任何值。图 5-3 所示为用模糊集合表示的重感冒患者集合。图中使用颜色深浅来表示不同体温隶属于重感冒集合的程度，从中可以看出：体温为 94℉的患者肯定不是重感冒患者，而体温为 110℉的患者一定是重感冒患者，而体温介于之间的患者仅在一定程度上趋向于重感冒，这样就引出了模糊集合的概念。

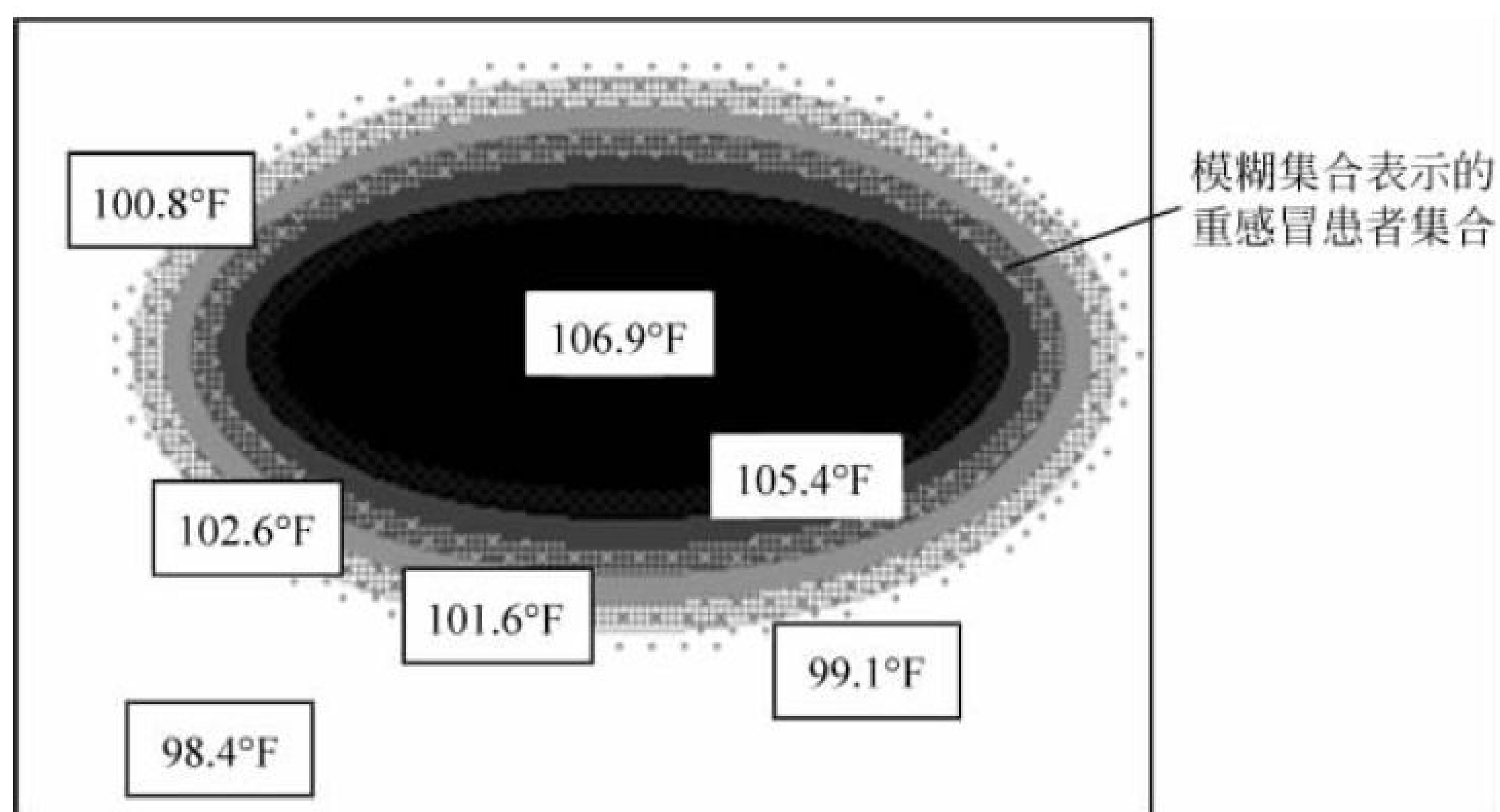


图 5-3 用模糊集合表示的重感冒患者集合

定义 1：设 U 是论域，U 上的一个实值函数用 $\mu_A(x)$ 来表示，即 $\mu_A(x): x\to[0,1]$，则称集合 A 为论域 U 上的模糊集合或模糊子集；对于 $x\in A$，$\mu_A(x)$ 称为 x 对 A 的隶属度，而 $\mu_A(x)$ 称为隶属度函数。

这样，对于论域 U 的一个元素 x 和 U 上的一个模糊子集 A，我们不再是简单地问 x 绝对属于还是不属于 A，而是问 x 在多大程度上属于 A。隶属度 $\mu_A(x)$ 正是 x 属于 A 的程度

的数量指标。若：

$\mu_A(x)=1$，则认为 x 完全属于 A；

$\mu_A(x)=0$，则认为 x 完全不属于 A；

$0<\mu_A(x)<1$，则认为 x 在 $\mu_A(x)$ 程度上属于 A。

这时，在完全属于 A 和不完全属于 A 的元素之间，呈现出中间过渡状态，或者叫作连续变化状态，这就是我们所说的 A 的外延表现出不分明的变化层次，或者表现出模糊性。

此时，根据模糊的定义就可以在患者体温和重感冒之间做出如下分析：

$$\mu_A(94℉)=0, \mu_A(100℉)=0.1, \mu_A(106℉)=0.9$$

$$\mu_A(96℉)=0, \mu_A(102℉)=0.35, \mu_A(108℉)=1$$

$$\mu_A(98℉)=0, \mu_A(104℉)=0.65, \mu_A(110℉)=1$$

为了清晰地判断患者的体温是否已达到重感冒的程度，或者患者的体温属于重感冒的程度，可以使用图 5-4 所示的隶属度函数表示。从图中可以看出，102℉的体温和 101.9℉的体温被评估为重感冒的程度是不同的，但它们之间的差别特别小。这样的表示方法更接近人的思维习惯。

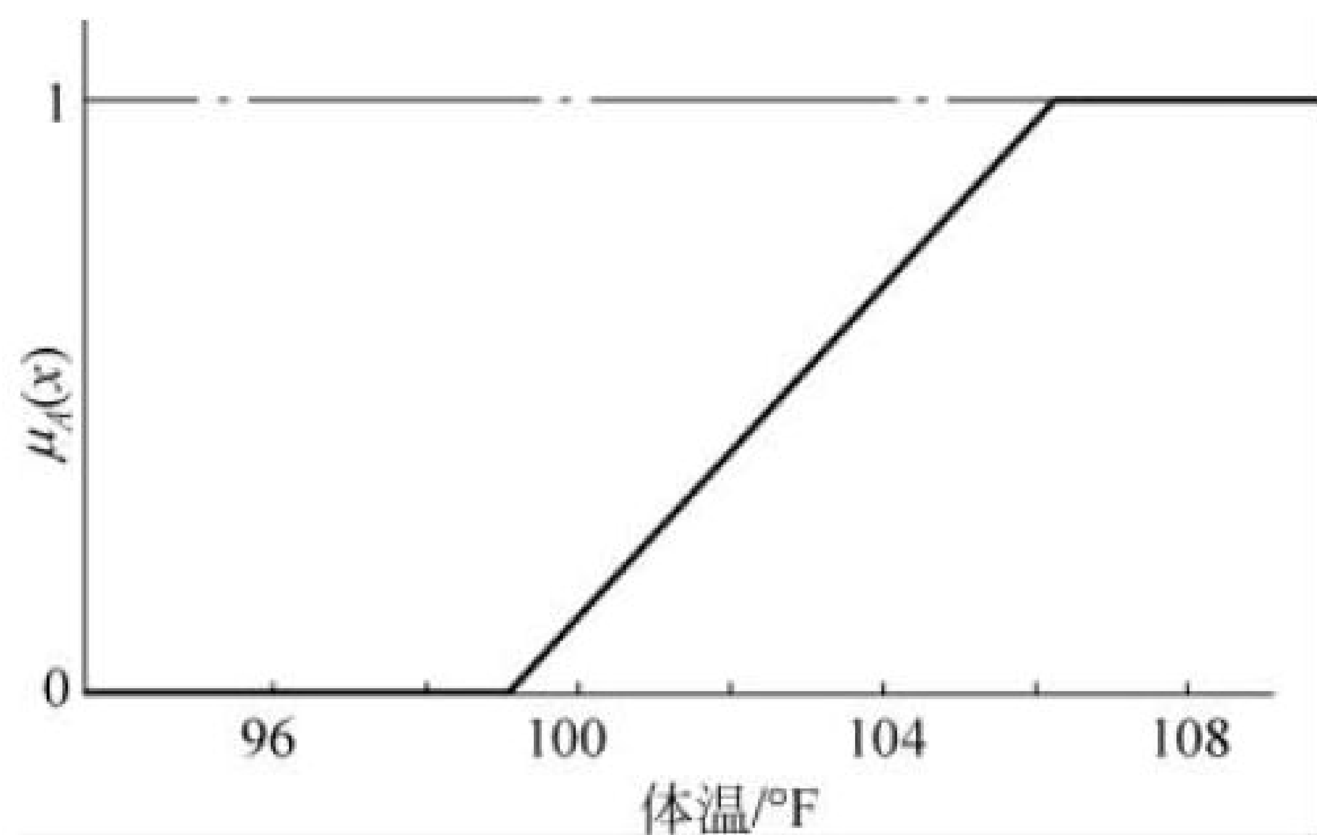


图 5-4　属于重感冒的隶属度函数图

综上所述，可以得出这样的结论：模糊集是传统集合的推广；传统指标函数的 $\mu=0$ 和 $\mu=1$ 刚好是模糊集合的特例。

模糊集合 A 是一个抽象的东西，而函数 $\mu_A(x)$ 则是具体的，即重感冒患者的模糊集合很难把握，因此只能通过体温属于重感冒的隶属度函数来认识和掌握集合 A。

常用的模糊集合有以下 3 种表示方法。

(1) 序偶表示法：$A=\{(x,\mu_A(x)), x\in U\}$。

(2) Zadeh 表示法：当论域 U 为有限集，即 $U=\{x_1,x_2,\cdots,x_n\}$ 时，U 上的模糊集合 A 可表示为 $A=\{\mu_A(x_1)/x_1+\mu_A(x_2)/x_2,\cdots,\mu_A(x_n)/x_n\}$；当论域 U 为无限集时，记作 $A=\int_x \mu_A(x)/x$。

(3) 隶属度函数解析式表示法：当论域 U 上为实数集 **R** 上的某区间时，直接给出模糊集合隶属度函数的解析式，是使用十分方便的一种表达形式。如 Zadeh 给出“年轻”的模糊集合 Y，其隶属度函数为

$$\mu_Y(x)=\begin{cases}1, & 0\leqslant x\leqslant 25\\ \left[1+\left(\dfrac{x-25}{5}\right)^2\right]^{-1}, & 25<x\leqslant 100\end{cases} \tag{5-1}$$

Zadeh 给出“年轻”的模糊集合 Y 的隶属度函数图如图 5-5 所示。

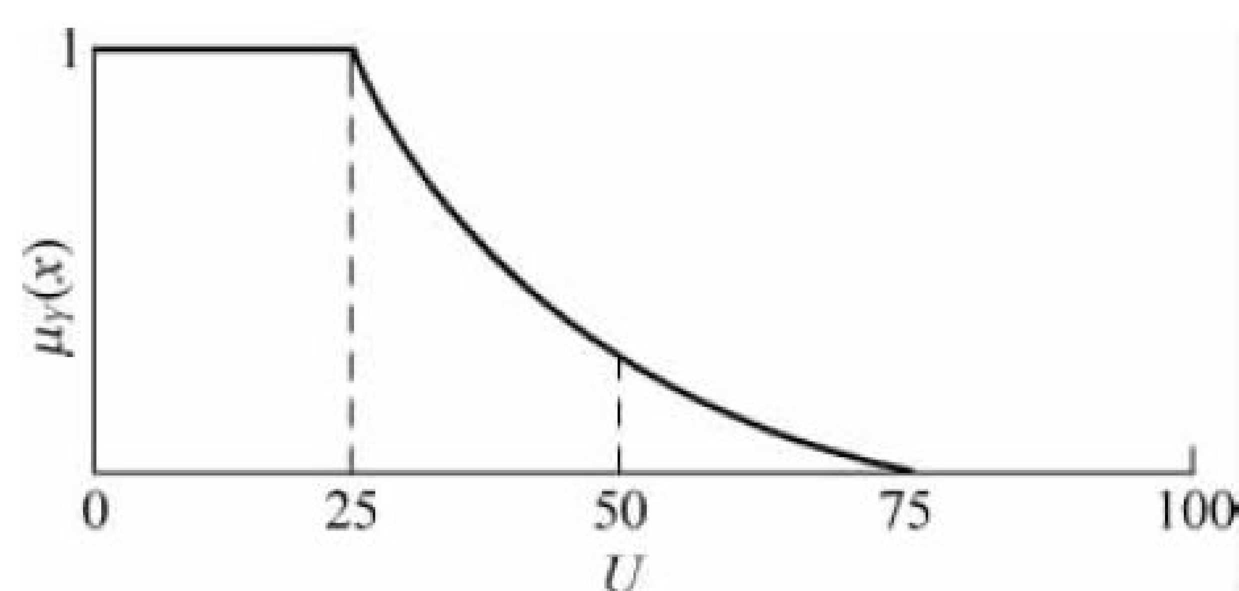


图 5-5 Zadeh 给出“年轻”的模糊集合 Y 的隶属度函数图

为了书写方便，模糊集合可写成 F 集(Fuzzy 的首个大写字母)；F 集合 A 的隶属度函数 $\mu_A(x)$简记为 $A(x)$。

定义 2：设 A 和 B 均为 U 上的模糊集，如果对所有的 x，即 $\forall x \in U$，均有 $\mu_A(x)=\mu_B(x)$，则称 A 和 B 相等，记作 $A=B$。

定义 3：设 A 和 B 均为 U 上的模糊集，如果 $\forall x \in U$，均有 $\mu_A(x) \leqslant \mu_B(x)$，则称 B 包含 A，或者称 A 是 B 的子集，记作 $A \subseteq B$。

定义 4：设 A 为 U 中的模糊集，如果对 $\forall x \in U$，均有 $\mu_A(x)=0$，则称 A 为空集，记作$\varnothing$。

定义 5：设 A 为 U 中的模糊集，如果对 $\forall x \in U$，均有 $\mu_A(x)=1$，则称 A 为全集，记作 Ω。

显然，$\varnothing \leqslant A \leqslant \Omega$。

对于同样的背景，我们可能有多个主观判断，如图 5-6 所示。其中，low 曲线为“体温低于正常体温”隶属度函数图；normal 曲线为“正常体温”隶属度函数图；raised 曲线为“体温高于正常体温但低于重感冒患者体温”隶属度函数图；strong_fever 曲线为“重感冒患者体温”的隶属度函数图。

定义 6：论域 U 上的模糊集 A 包含了 U 中所有在 A 上具有非零隶属度值的元素，即 $\mathrm{supp}(A)=\{x \in U \mid \mu_A(x)>0\}$，式中 $\mathrm{supp}(A)$表示模糊集合 A 的支集。模糊集的支集是经典集合。

定义 7：如果一个模糊集的支集是空的，则称该模糊集为空模糊集。

定义 8：如果模糊集合的支集仅包含 U 中的一个点，则称该模糊集为模糊单值。

定义 9：论域 U 上的模糊集 A 包含了 U 中所有在 A 上隶属度值为 1 的元素，即 $\mathrm{Ker}(A)=\{x \in U \mid \mu_A(x)=1\}$，式中 $\mathrm{Ker}(A)$表示模糊集合 A 的核。模糊集的核也是经典集合。

定义 10：如果模糊集的隶属度函数达到其最大值的所有点的均值是有限值，则将该均值定义为模糊集的中心；如果该均值为正(负)无穷大，则将该模糊集的中心定义为所有达到最大隶属度值的点中的最小(最大)点的值，如图 5-7 所示。

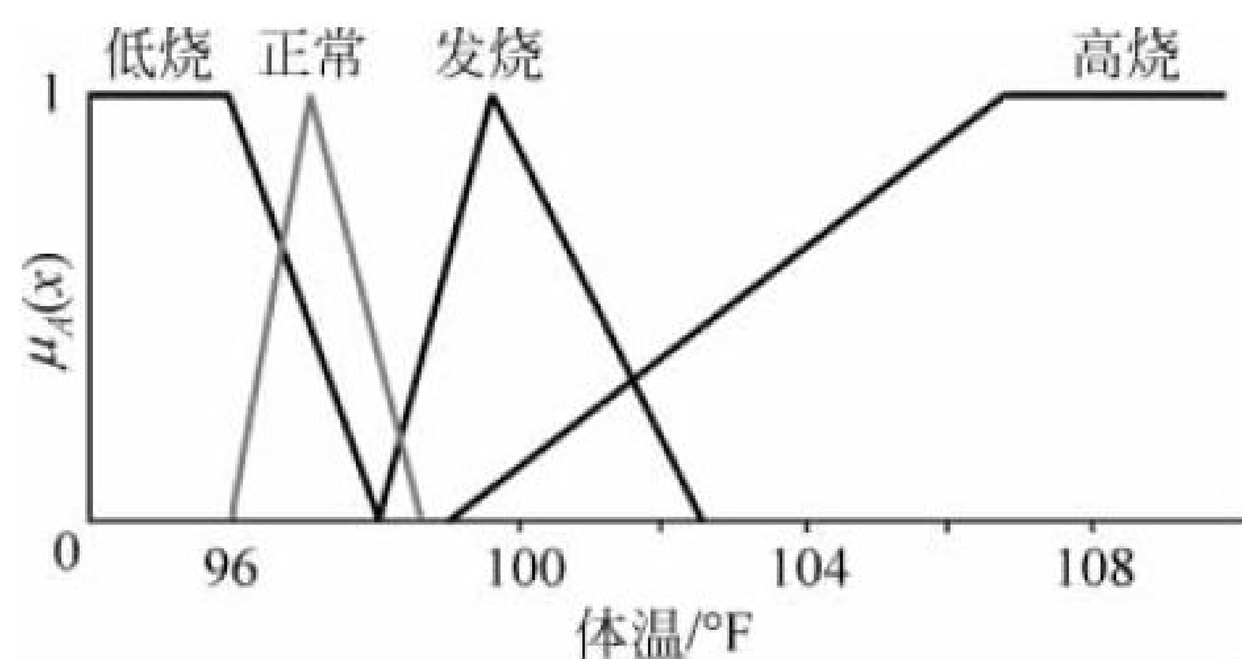


图 5-6 同样背景的不同主观判断隶属度函数图

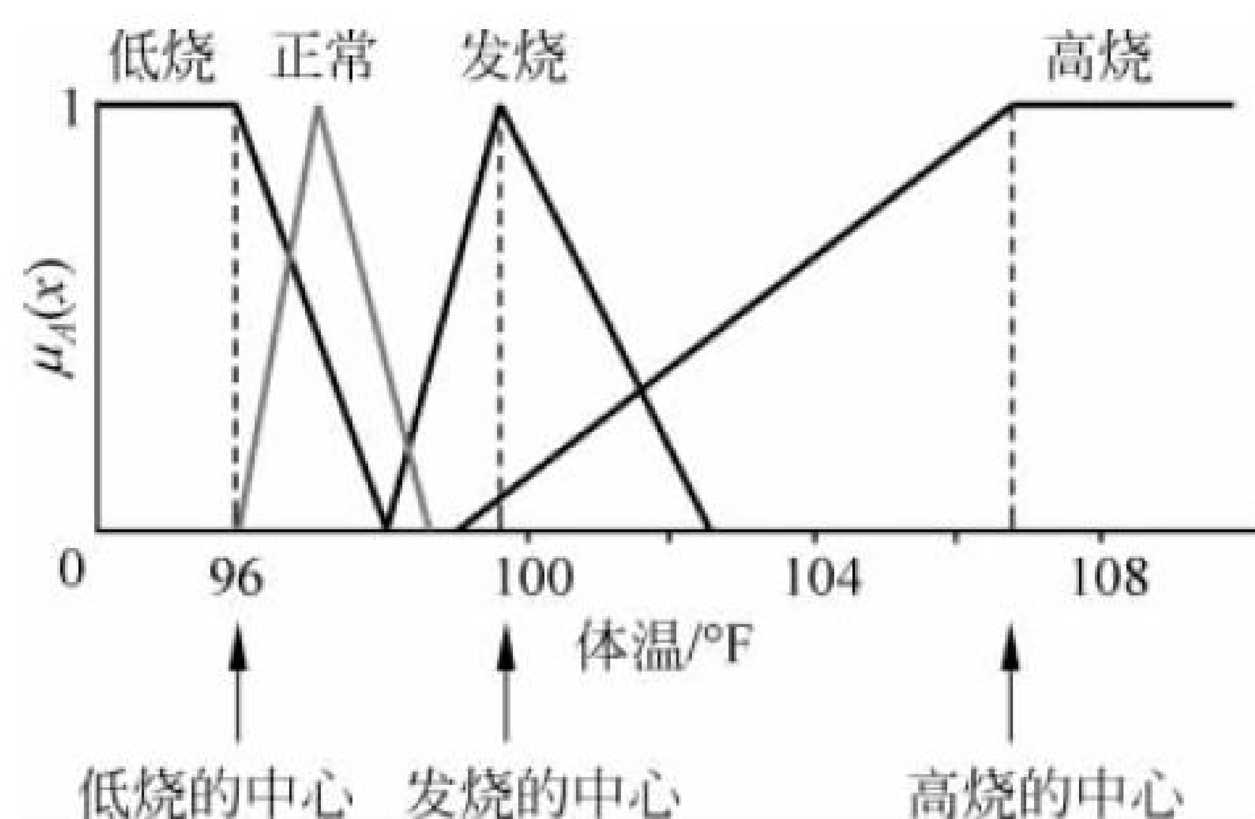


图 5-7 一些典型模糊集的中心

定义 11：一个模糊集的交叉点就是 U 中隶属于 A 的隶属度值等于 0.5 的值。

定义 12：模糊集的高度是指任意点所达到的最大隶属度值。图 5-8 所示的隶属度函数的高度均等于 1。

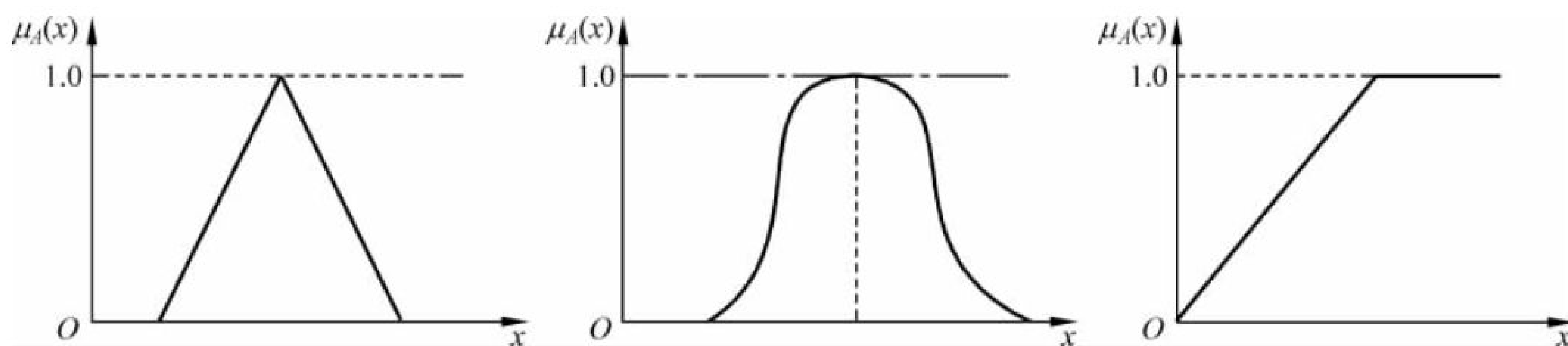


图 5-8　三角形、钟形及 S 形隶属度函数图

如果一个模糊集的高度等于 1，则称为标准模糊集。

定义 13：设 A 是以实数 **R** 为论域的模糊集，其隶属度函数为 $\mu_A(x)$，如果对任意实数 $a<x<b$，都有

$$\mu_A(x) \geqslant \min(\mu_A(a), \mu_A(b)) \quad a,b,x \in \mathbf{R}$$

则称 A 是一个凸模糊集。

与凸模糊集相对的为非凸模糊集，凸模糊集与非凸模糊集的示意图如图 5-9 所示。

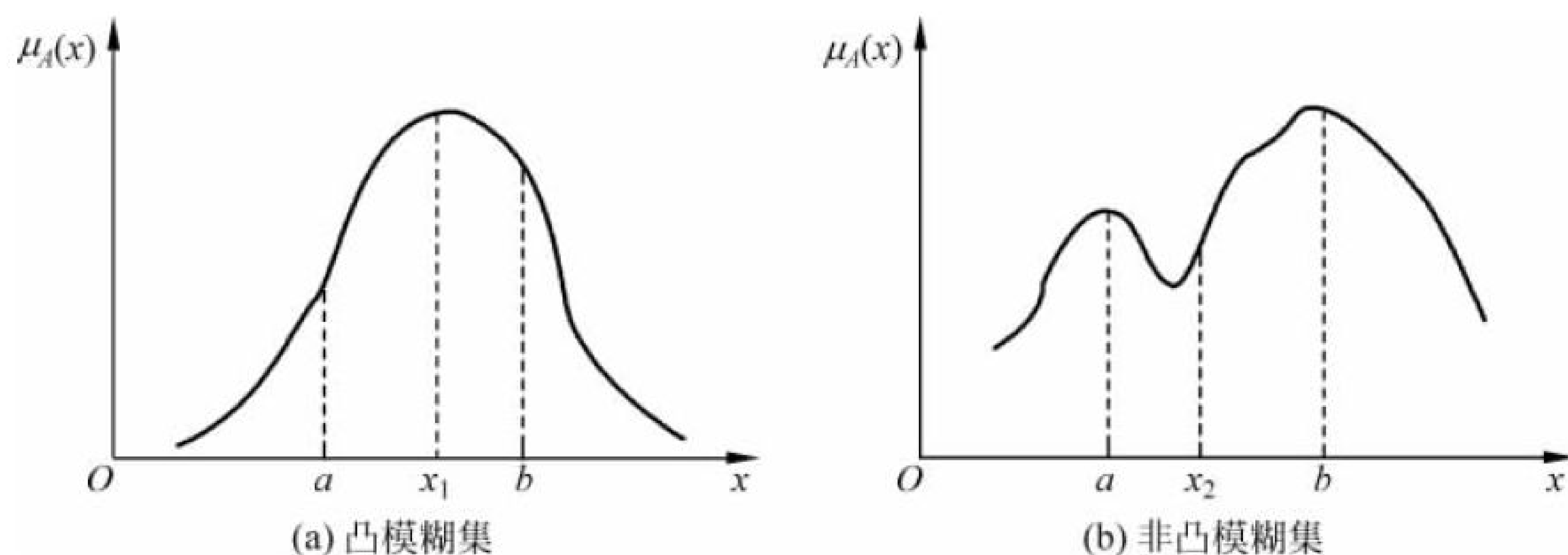


图 5-9　凸模糊集与非凸模糊集示意图

5.2.3　隶属度函数

经典集合使用特征函数来描述，模糊集合使用隶属度函数作定量描述。因此，隶属度函数是模糊集合的核心。定义一个模糊集合就是定义论域中各个元素对该模糊集合的隶属度。

经典集合的特征函数的值域为集合{0,1}，模糊集合的隶属度函数的值域为区间[0,1]。隶属度函数是特征函数的扩展和一般化。

从使用模糊集合表示的重感冒患者一例中可以看出，体温隶属于重感冒的程度需要人为确定，即模糊集合隶属度函数是人为主观定义的一种函数。从隶属度函数的确定过程看，隶属度函数本质上说应该是客观的，但每个人对于同一个模糊概念的认识理解又有差异。因此，隶属度函数的确定又带有主观性。所以，隶属度函数包含了太多的人的主观意志，从而很难使用统一的方法确定隶属度函数。

对于同一个模糊概念，不同的人会建立不完全相同的隶属度函数，尽管形式不完全相

同，只要能反映同一模糊概念，在解决和处理实际模糊信息的问题中仍然殊途同归，这是因为隶属度函数是人们长期实践经验的总结，可以反映客观实际，还具有一定的客观性、科学性和准确性。至今为止，确定隶属度函数的方法大多依靠经验、实践和实验数据，经常使用的确定隶属度函数的方法有以下 4 种。

1. 模糊统计法

模糊统计法的基本思想：对论域 U 上的一个确定元素 x_1 是否属于论域上的一个可变动的经典集合 B 做出清晰的判断。对于不同的试验者，经典集合 B 可以有不同的边界，但它们都对应于同一个模糊集 A。在每次统计中，x_1 是固定的，B 的值是可变的，作 n 次试验，其模糊统计可按下式进行计算

$$x_1 \text{ 对 } A \text{ 的隶属频率} = \frac{x_1 \in A \text{ 的次数}}{\text{试验总次数 } n} \tag{5-2}$$

随着 n 的增大，隶属频率也会趋向稳定，这个稳定值就是 x_1 对 A 的隶属度值。这种方法较直观地反映了模糊概念中的隶属程度，但其计算量较大。

2. 例证法

例证法的主要思想是从已知有限个 $\mu_A(x)$的值，来估计论域 U 上的模糊子集 A 的隶属度函数。如论域 U 代表全体人类，A 是“高个子的人”，显然 A 是一个模糊子集。为了确定 μ_A，先确定一个高度值 h，然后选定几个语言真值(即一句话的真实程度)中的一个来回答某人是否算“高个子”。语言真值可分为“真的”“大致真的”“似真似假”“大致假的”和“假的”五种情况，并且分别用数字 1、0.75、0.5、0.25、0 来表示这些语言真值。对 n 个不同高度 h_1，h_2，…，h_n都作同样的询问，就可以得到 A 的隶属度函数的离散表示。

3. 专家经验法

专家经验法是根据专家的实际经验给出模糊信息的处理算式或相应权系数值来确定隶属度函数的一种方法。在许多情况下，首先确定粗略的隶属度函数，然后再通过“学习”和实践检验逐步修改和完善，而实际效果正是检验和调整隶属度函数的依据。

4. 二元对比排序法

二元对比排序法是一种较实用的确定隶属度函数的方法。它通过对多个事物之间的两两对比来确定某种特征下的顺序，由此来决定这些事物对该特征的隶属度函数的大体形状。二元对比排序法根据对比测度不同，可分为相对比较法、对比平均法、优先关系定序法和相似优先对比法等。

在实际工作中，为了兼顾计算和处理的简便性，经常把使用不同方法得出的数据近似地表示成常用的解析函数形式，构成常用的隶属度函数。

(1) 三角形：三角形隶属度曲线对应的数学表达式为

$$f(x,a,b,c) = \begin{cases} 0, & x \leqslant a \\ \dfrac{x-a}{b-a}, & a \leqslant x \leqslant b \\ \dfrac{c-x}{c-b}, & b \leqslant x \leqslant c \\ 0, & x \geqslant c \end{cases} \tag{5-3}$$

（2）钟形：钟形隶属度曲线对应的数学表达式为

$$f(x,a,b,c)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{5-4}$$

式中，c 决定函数的中心位置；a、b 决定函数的形状。

（3）高斯：高斯隶属度曲线对应的数学表达式为

$$f(x,\sigma,c)=\mathrm{e}^{-\frac{(x-c)^2}{2\sigma^2}} \tag{5-5}$$

式中，c 决定函数的中心位置；σ 决定函数曲线的宽度。

（4）梯形：梯形隶属度曲线对应的数学表达式为

$$f(x,a,b,c,d)=\begin{cases}0, & x\leqslant a\\ \dfrac{x-a}{b-a}, & a\leqslant x\leqslant b\\ 1, & b\leqslant x\leqslant c\\ \dfrac{d-x}{d-c}, & c\leqslant x\leqslant d\\ 0, & x\geqslant d\end{cases} \tag{5-6}$$

式中 $a\leqslant b, c\leqslant d$。

（5）Sigmoid 形：Sigmoid 形隶属度曲线对应的数学表达式为

$$f(x,a,c)=\frac{1}{1+\mathrm{e}^{-a(x-c)}} \tag{5-7}$$

式中，a、c 决定函数的形状。

5.3 模糊集合的运算

和经典集合一样，模糊集合也包含“交”“并”和“补”运算。比如选购衣服，到底选择哪件衣服呢？花色较好、样式不错、价格也合理的衣服应该是理想的选择，这就应用到了模糊集合的“交”运算；比如点菜，要荤素搭配，此时就需要进行模糊集合的“并”运算；比如租一处面积不大的房子，则可求取“面积大的房子”的集合的补集。可见，通过对模糊集合做运算，可得到更多衍生结论。

5.3.1 模糊集合的基本运算

定义 14：设 A、B 为 U 中的两个模糊集。隶属度函数分别为 $\mu_A(x)$ 和 $\mu_B(x)$，则模糊集 A 和 B 的并集 $A\cup B$、交集 $A\cap B$ 和补集 A^C 的运算可通过它们的隶属度函数来定义：

并集：$\mu_{A\cup B}(x)=\mu_A(x)\vee\mu_B(x)$，其中 $\vee$ 表示两者比较后取大值。

交集：$\mu_{A\cap B}(x)=\mu_A(x)\wedge\mu_B(x)$，其中 $\wedge$ 表示两者比较后取小值。

补集：$\mu_{A^C}(x)=1-\mu_A(x)$。

模糊集合的基本运算可用图 5-10 所示曲线加以说明。

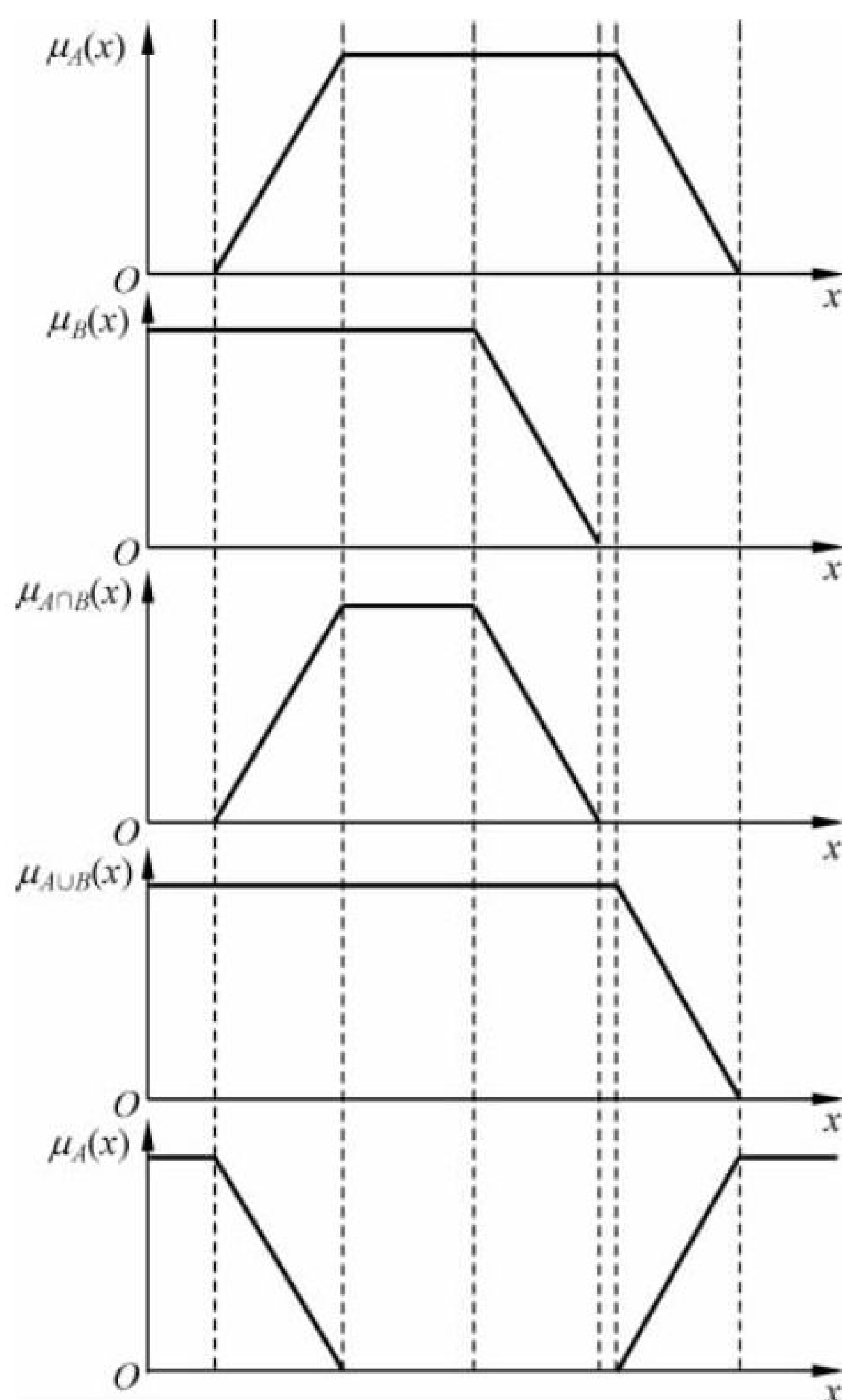


图 5-10 模糊集合的基本运算示意图

例 1：设 $U=\{u_1,u_2,u_3,u_4,u_5\}$，若 $A,B\in F(U)$，$A=\frac{0.2}{u_1}+\frac{0.7}{u_2}+\frac{1}{u_3}+\frac{0.5}{u_5}$，$B=\frac{0.5}{u_1}+\frac{0.3}{u_2}+\frac{0.1}{u_4}+\frac{0.7}{u_5}$，求 $A\cup B$、$A\cap B$、A^c、$A\cup A^c$ 和 $A\cap A^c$。

解：
$$A\cup B=\frac{0.2\vee 0.5}{u_1}+\frac{0.7\vee 0.3}{u_2}+\frac{1\vee 0}{u_3}+\frac{0\vee 0.1}{u_4}+\frac{0.5\vee 0.7}{u_5}$$
$$=\frac{0.5}{u_1}+\frac{0.7}{u_2}+\frac{1}{u_3}+\frac{0.1}{u_4}+\frac{0.7}{u_5}$$

$$A\cap B=\frac{0.2\wedge 0.5}{u_1}+\frac{0.7\wedge 0.3}{u_2}+\frac{1\wedge 0}{u_3}+\frac{0\wedge 0.1}{u_4}+\frac{0.5\wedge 0.7}{u_5}$$
$$=\frac{0.2}{u_1}+\frac{0.3}{u_2}+\frac{0.5}{u_5}$$

$$A^c=\frac{1-0.2}{u_1}+\frac{1-0.7}{u_2}+\frac{1-1}{u_3}+\frac{1-0}{u_4}+\frac{1-0.5}{u_5}$$
$$=\frac{0.8}{u_1}+\frac{0.3}{u_2}+\frac{1}{u_4}+\frac{0.5}{u_5}$$

$$A\cup A^c=\frac{0.2\vee 0.8}{u_1}+\frac{0.7\vee 0.3}{u_2}+\frac{1\vee 0}{u_3}+\frac{0\vee 1}{u_4}+\frac{0.5\vee 0.5}{u_5}$$
$$=\frac{0.8}{u_1}+\frac{0.7}{u_2}+\frac{1}{u_3}+\frac{1}{u_4}+\frac{0.5}{u_5}\qquad\text{（不是全集）}$$

$$A \cap A^c = \frac{0.2 \wedge 0.8}{u_1} + \frac{0.7 \wedge 0.3}{u_2} + \frac{1 \wedge 0}{u_3} + \frac{0 \wedge 1}{u_4} + \frac{0.5 \wedge 0.5}{u_5}$$

$$= \frac{0.2}{u_1} + \frac{0.3}{u_2} + \frac{0.5}{u_5} \qquad \text{(不是空集)}$$

需要注意的是，在经典集合中，集合 A 和它的补集 A^C 的并集为全集，集合 A 和它的补集 A^C 的交集为空集，但在模糊集合论中却没有这样的结论。这里用图示的方法加以理解，如图 5-11 所示。

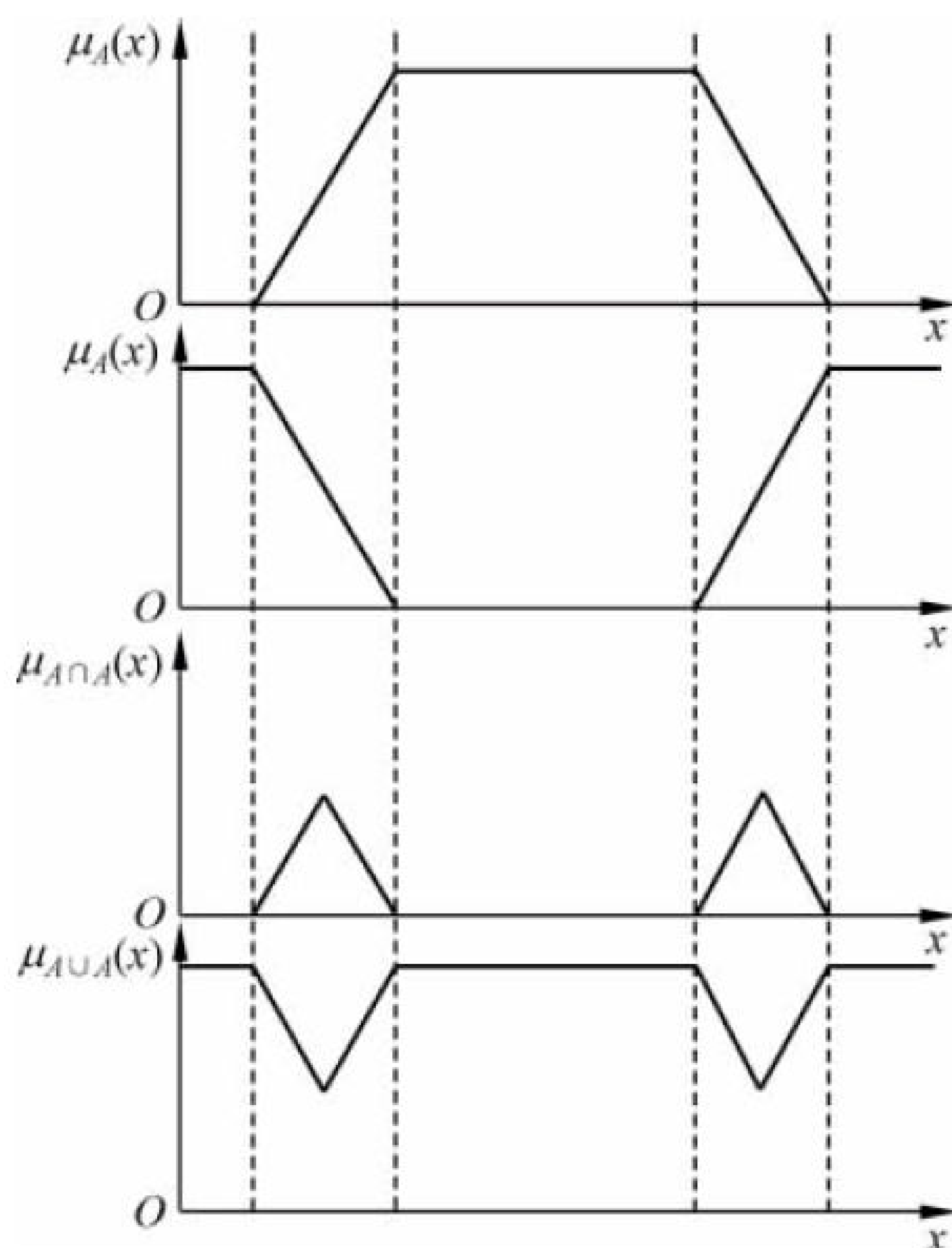


图 5-11 模糊集合 A 和它的补集 A^C 的交集及并集

虽然模糊集合的基本运算与经典集合的基本运算有许多相似之处，但是经典运算是对论域中元素的归属做新的划分，而模糊集合的运算是对论域中的元素对于模糊集合的隶属度做新的调整。

定义 15：设 A、B 为 U 中的两个模糊集，隶属度函数分别为 $\mu_A(x)$ 和 $\mu_B(x)$，则模糊集 A 和 B 的代数积 $(A \cdot B)$、代数和 $(A+B)$、有界和 $(A \oplus B)$、有界积 $(A \odot B)$ 可通过它们的隶属度函数定义如下：

代数积：$\mu_{A\cdot B}(x) = \mu_A(x) \times \mu_B(x)$。

代数和：$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$。

有界和：$\mu_{A\oplus B}(x) = (\mu_A(x) + \mu_B(x)) \wedge 1 = \min(A(x)+B(x), 1)$。

有界积：$\mu_{A\odot B}(x) = (\mu_A(x) + \mu_B(x)) \vee 0 = \max(0, A(x)+B(x)-1)$。

式中，min 为取最小值运算；max 为取最大值运算。图 5-12 所示为模糊集合代数和、代数积、有界和、有界积的图示。

5.3.2 模糊集合的基本运算规律

两个模糊集合的运算，实际上就是逐点对其隶属度作相应的运算。模糊集合 $A, B, C \in F(U)$ 的并、交、补运算满足以下性质：

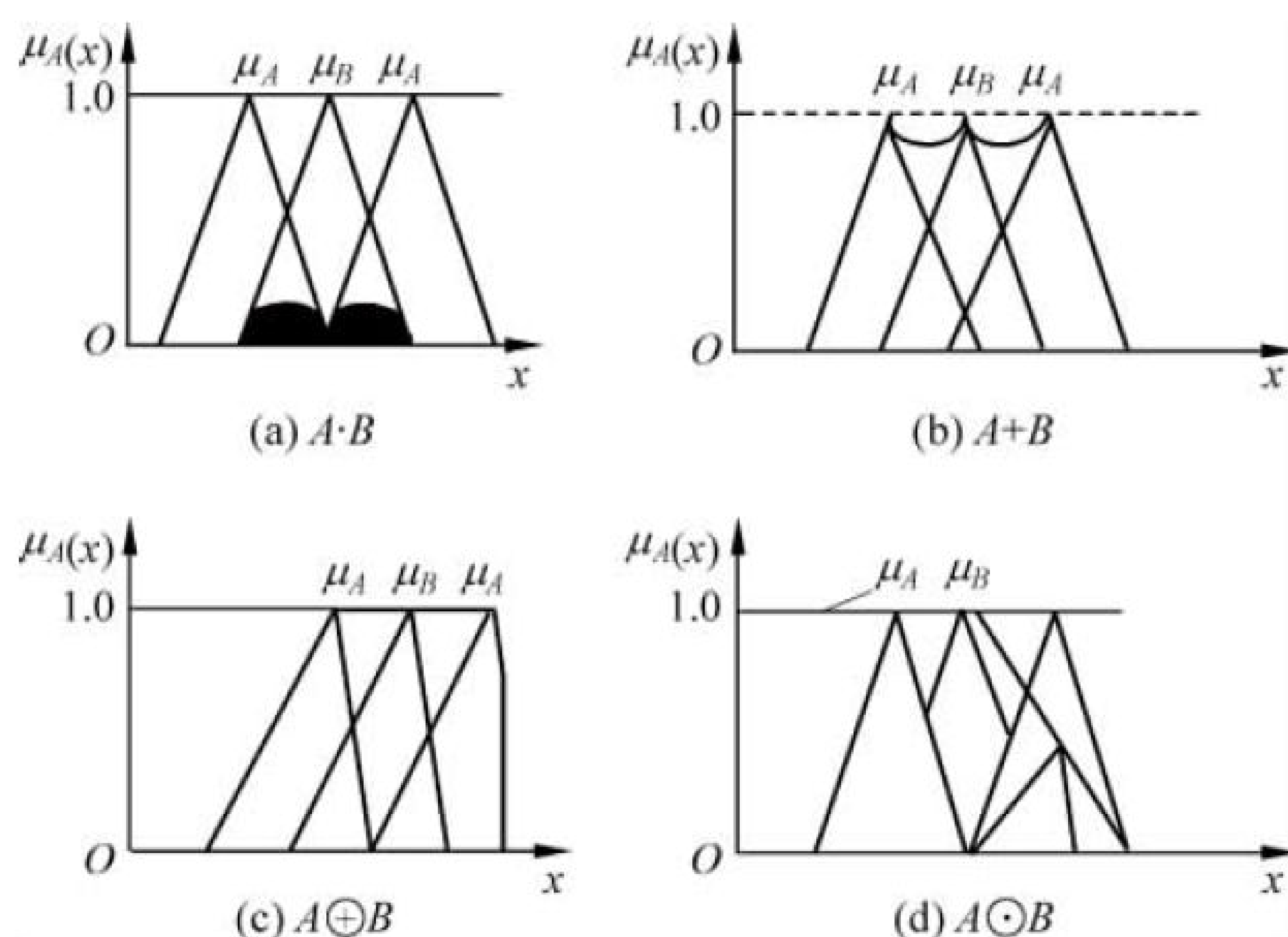


图 5-12 模糊集合代数和、代数积、有界和、有界积的图示

(1) 幂等律：$A\cup A=A$，$A\cap A=A$。

(2) 交换律：$A\cup B=B\cup A$，$A\cap B=B\cap A$。

(3) 结合律：$(A\cup B)\cup C=A\cup(B\cup C)$，$(A\cap B)\cap C=A\cap(B\cap C)$。

(4) 吸收律：$(A\cap B)\cup A=A$，$(A\cup B)\cap A=A$。

(5) 分配律：$A\cap(B\cup C)=(A\cap B)\cup(A\cap C)$，$A\cup(B\cap C)=(A\cup B)\cap(A\cup C)$。

(6) 零一律：$A\cup U=U$；$A\cap U=A$；$A\cup\varnothing=A$；$A\cap\varnothing=\varnothing$。

(7) 复原律：$(A^c)^c=A$。

(8) 摩根律：$(A\cup B)^c=A^c\cap B^c$；$(A\cap B)^c=A^c\cup B^c$。

模糊集合与经典集合的一个显著的不同之处是：模糊集合的并、交、补运算一般不满足补余律，即 $A\cup A^c\neq U$，$A\cap A^c\neq\varnothing$。

5.3.3 模糊集合与经典集合的联系

当医生诊断发热患者是否为重感冒患者时，就需要对“重感冒”这一模糊概念有明确的认识和判断。当判断某个发热患者对“重感冒”集合的明确归属时，这就要求模糊集合与经典集合可以依据某种法则相互转换。模糊集合与经典集合之间的联系可通过 λ-截集和分解定理表示。

定义 16：一个模糊集的 λ-截集是指包含 U 中所有隶属于 A 的、隶属度值大于等于 λ 的元素，即 $A_\lambda=\{x\in U\,|\,\mu_A(x)\geqslant\lambda\}$。$\lambda$-截集示意图如图 5-13 所示。

其中，图 5-13(b)所示为 λ_1-截集的特征函数描述；图 5-13(c)所示为 λ_2-截集的特征函数描述。从图可以看出，λ-截集是经典集合。对于 λ-截集，我们可以这样理解：模糊集合 A 本身是一个没有确定边界的集合，但是如果约定，凡 x 对 A 的隶属度达到或超过某个 λ 水平者才算是 A 的成员，那么模糊集合 A 就变成了普通集合 A_λ。

当 $\lambda=1$ 时，得到最小水平截集 A_1，即模糊集 A 的核；当 $\lambda=0^+$ 时，得到最大水平的截集，即模糊集 A 的支集。

若模糊集 A 的核非空，则称 A 为正规模糊集；否则，称 A 为非正规模糊集。

定义 17：设 A 是普通集合，$\lambda\in[0,1]$，做数量积运算，得到一个特殊的模糊集 λ_A，其隶属度函数为

$$\mu_{\lambda A}(x)=\begin{cases}\lambda, & x\in A\\ 0, & x\notin A\end{cases}\tag{5-8}$$

分解定理：设 A 为论域 x 上的模糊集合，A_λ 是 A 的截集，则 $A=\bigcup_{\lambda\in[0,1]}\lambda A_\lambda$。

分解定理可用图 5-14 表示。

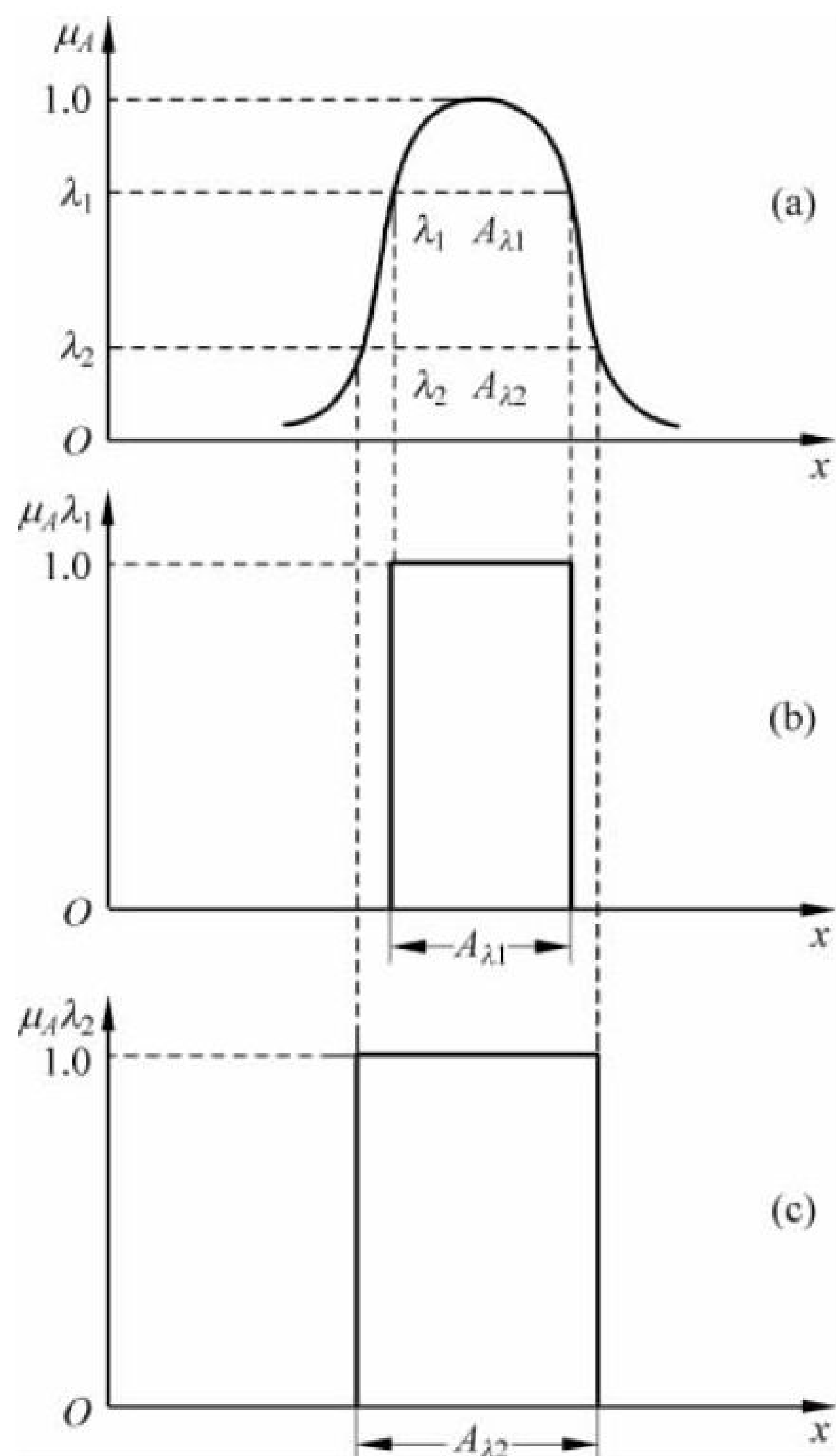


图 5-13　λ-截集示意图

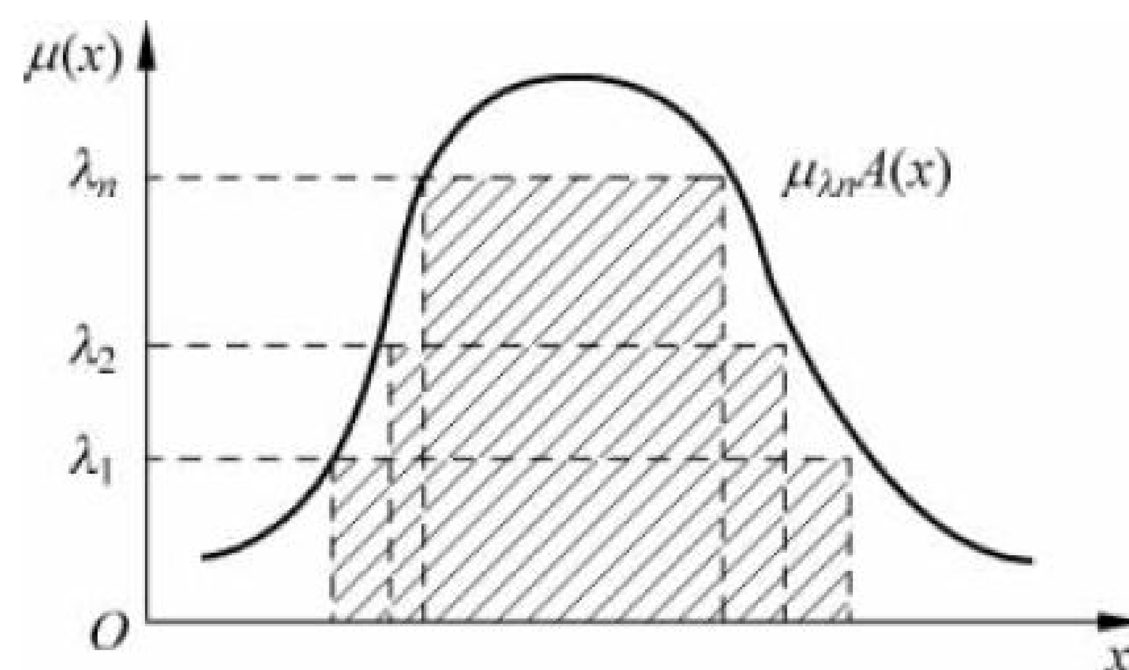


图 5-14　分解定理示意图

如果 λ 遍取区间[0,1]中的实数，按照模糊集合求并运算的法则，$\bigcup_{\lambda\in[0,1]}\lambda A_\lambda$ 恰好取各 λ 点隶属度函数的最大值，将这些点连成一条曲线，正是 A 的隶属度函数 $\mu_A(x)$。

例 2：设 $A=\frac{0.2}{u_1}+\frac{0.7}{u_2}+\frac{1}{u_3}+\frac{0.6}{u_4}+\frac{0.5}{u_5}$，则

$$A_{0.2}=\{u_1,u_2,u_3,u_4,u_5\}$$

$$0.2A_{0.2}=\frac{0.2}{u_1}+\frac{0.2}{u_2}+\frac{0.2}{u_3}+\frac{0.2}{u_4}+\frac{0.2}{u_5}$$

$$A_{0.5}=\{u_2,u_3,u_4,u_5\}$$

$$0.5A_{0.5}=\frac{0.5}{u_2}+\frac{0.5}{u_3}+\frac{0.5}{u_4}+\frac{0.5}{u_5}$$

$$A_{0.6}=\{u_2,u_3,u_4\}$$

$$0.6A_{0.6}=\frac{0.6}{u_2}+\frac{0.6}{u_3}+\frac{0.6}{u_4}$$

$$A_{0.7}=\{u_2,u_3\}$$

$$0.7A_{0.7}=\frac{0.7}{u_2}+\frac{0.7}{u_3}$$

$$A_1=\{u_3\}$$

$$1\times A_1=\frac{1}{u_3}$$

则

$$A=\bigcup_{\lambda\in[0,1]}\lambda A_\lambda=0.2A_{0.2}\cup 0.5A_{0.5}\cup 0.6A_{0.6}\cup 0.7A_{0.7}\cup A_1$$

$$=\frac{0.2}{u_1}+\frac{0.7}{u_2}+\frac{1}{u_3}+\frac{0.6}{u_4}+\frac{0.5}{u_5}$$

A 是模糊集合，A_λ 是经典集合，它们之间的联系和转化由分解定理用数学语言表达出来。这个定理也说明了模糊性的成因，大量甚至无限多的清晰事物叠加在一起，总体上就形成了模糊事物。

5.4 模糊关系与模糊关系的合成

事物都是普遍联系的，集合论中的“关系”抽象地刻画了事物“精确性”的联系，而模糊关系则从更深刻的意义上表现了事物间更广泛的联系。从某种意义上讲，模糊关系的抽象更接近人的思维方式。

5.4.1 模糊关系的基本概念

元素间的联系不是简单的有或无，而是不同程度的隶属关系，因此这里引入模糊关系。

定义 18：给定集合 X 和 Y，由全体 $(x,y)(x\in X,y\in Y)$ 组成的集合叫作 X 和 Y 的笛卡儿积（或称直积），记作 $X\times Y$，$X\times Y=\{(x,y)\mid(x\in X,y\in Y)\}$。

例 3：国际上常用的人的体重计算公式为标准体重＝(身高(cm)－100)×0.9(kg)，那么实际身高与实际体重之间就存在模糊关系。如果身高的集合 $X=\{150,155,160,165\}$，体重的集合 $Y=\{45,49.5,54,58.5\}$，则

$$\begin{aligned}X\times Y=\{&(150,45),(150,49.5),(150,54),(150,58.5),\\&(155,45),(155,49.5),(155,54),(155,58.5),\\&(160,45),(160,49.5),(160,54),(160,58.5),\\&(165,45),(165,49.5),(165,54),(165,58.5),\}\end{aligned}$$

定义 19：存在集合 X 和 Y，它们的笛卡儿积 $X\times Y$ 的一个子集 R 叫作 X 到 Y 的二元关系，简称关系，$R\subseteq X\times Y$。序偶 (x,y) 是笛卡儿积 $X\times Y$ 的元素，它是无约束的组对。若给组对以约束，便体现了一种特定的关系。受到约束的序偶则形成了 $X\times Y$ 的一个子集。

- 若 $X=Y$，则称 R 是 X 中的关系。
- 如果 $(x,y)\in R$，则称 X 和 Y 有关系 R，记作 xRy。

例 4：身高集合 $X=\{150,155,160,165\}$，体重集合 $Y=\{45,49.5,54,58.5\}$，根据国际上常用的人的体重计算公式：标准体重＝(身高(cm)－100)×0.9(kg)，则对应身高“非标准”体重时，可采用模糊关系表示身高、体重与标准体重之间的关系，如表 5-1 所示。

表 5-1　身高、体重与标准体重的模糊关系表

$\mu_R(x,y)$		体重			
		45	49.5	54	58.5
身高	150	1	0.75	0.3	0
	155	0.75	1	0.75	0.3
	160	0.3	0.75	1	0.75
	165	0	0.3	0.75	1

◆ 如果 $(x,y)\notin R$，则称 X 和 Y 没有关系，记作 $x\bar{R}y$，也可用特征函数表示为

$$\mu_R(x,y)=\begin{cases}1, & (x,y)\in R\\ 0, & (x,y)\notin R\end{cases} \tag{5-9}$$

◆ 当 X 和 Y 都是有限集合时，关系可以用矩阵来表示，称关系矩阵。设 $X=\{x_1,x_2,\cdots,x_m\}$，$Y=\{y_1,y_2,\cdots,y_n\}$，则 $\boldsymbol{R}$ 可以表示为 $\boldsymbol{R}=[r_{ij}]$。其中 $r_{ij}=\mu_R(x_i,y_j)$；$i=1,2,\cdots,m$；$j=1,2,\cdots,n$。

例 5：身高集合 $X=\{150,155,160,165\}$，体重集合 $Y=\{45,49.5,54,58.5\}$，根据国际上常用的人的体重计算公式：标准体重＝(身高(cm)－100)×0.9(kg)，则对应身高“非标准”体重时，可采用模糊矩阵表示身高、体重与标准体重之间的关系。

$$\boldsymbol{R}=\begin{bmatrix}1 & 0.75 & 0.3 & 0\\ 0.75 & 1 & 0.75 & 0.3\\ 0.3 & 0.75 & 1 & 0.75\\ 0 & 0.3 & 0.75 & 1\end{bmatrix}$$

当矩阵中的元素等于 1 或等于 0 时，将这种矩阵称为布尔矩阵。

定义 20：设有集合 X、Y，如果有一对关系存在，对于任意 $x\in X$，有唯一的一个 $y\in Y$ 与之对应，我们就说，其对应关系是一个由 X 到 Y 的映射 f，记作

$$f: x\to Y$$

对任意 $x\in X$ 经映射后变成 $y\in Y$，则记作 $Y=f(x)$，此时 X 叫作 f 的定义域，而集合 $f(x)=\{f(x)|x\in X\}$ 称为 f 的值域，显然 $f(x)\subseteq Y$。

映射有时也叫作函数，但它通常是函数概念的推广。

定义 21：设 $f: X\to Y$

◆ 如果对每一 $x_1,x_2\in X$，$x_1\neq x_2$，则称 f 为单射(或称一一映射)。

◆ 如果 f 的值域是整个 Y，则称 f 为满射。

◆ 如果 f 既是单射的，又是满射的，则称 f 为一一对应的映射。

模糊关系是指笛卡儿积上的模糊集合，表示多个集合的元素间所具有的某种关系的程度。

定义 22：所谓 X、Y 两集合的笛卡儿积 $X\times Y=\{(x,y)|(x\in X,y\in Y)\}$ 中的一个模糊关系 $\boldsymbol{R}$，是指以 $X\times Y$ 为论域的一个模糊子集，序偶 (x,y) 的隶属度为 $\mu_R(x,y)$。$\mu_R(x,y)$ 在实轴的闭区间取值，它的大小反映了 (x,y) 具有关系 $\boldsymbol{R}$ 的程度。

由于模糊关系是一种模糊集合,因此模糊集合的相等、包含等概念对模糊关系同样具有意义。

设 X 是 m 个元素构成的有限论域,Y 是 n 个元素构成的有限论域。对于 X 到 Y 的一个模糊关系 $\boldsymbol{R}$,可以用一个 $m\times n$ 阶矩阵表示为

$$\boldsymbol{R}=\begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & & \vdots \\ r_{m1} & \cdots & r_{mn} \end{bmatrix} \tag{5-10}$$

或 $\boldsymbol{R}=[r_{ij}]$,$r_{ij}=\mu_R(x_i,x_j)$,$i=1,2,\cdots,m$,$j=1,2,\cdots,n$。

如果一个矩阵是模糊矩阵,它的每一个元素属于[0,1],则令

$$\boldsymbol{F}_{m\times n}=\{\boldsymbol{R}=[r_{ij}];0\leqslant r_{ij}\leqslant 1\} \tag{5-11}$$

$\boldsymbol{F}_{m\times n}$表示 $m\times n$ 阶模糊矩阵的全体。

在有限论域间,普通集合与布尔矩阵建立了一一对应的关系,模糊关系和模糊矩阵建立了一一对应的关系。

由于模糊矩阵本身是表示一个模糊关系的子集 R,因此根据模糊集的并、交、补运算的定义,模糊矩阵也可看作相应的运算。

设模糊矩阵 $\boldsymbol{R}$ 和 $\boldsymbol{Q}$ 是 $X\times Y$ 的模糊关系,$\boldsymbol{R}=[r_{ij}]_{m\times n}$,$\boldsymbol{Q}=[q_{ij}]_{m\times n}$,模糊集合的并、交、补运算为

模糊矩阵并运算:$\boldsymbol{R}\cup\boldsymbol{Q}=[r_{ij}\vee q_{ij}]_{m\times n}$

模糊矩阵交运算:$\boldsymbol{R}\cap\boldsymbol{Q}=[r_{ij}\wedge q_{ij}]_{m\times n}$

模糊矩阵补运算:$\boldsymbol{R}^C=[1-r_{ij}]_{m\times n}$

如果 $r_{ij}\leqslant q_{ij}$,$i=1,2,\cdots,m$,$j=1,2,\cdots,n$,则称 $\boldsymbol{R}$ 被模糊矩阵 $\boldsymbol{S}$ 包含,记为 $\boldsymbol{R}\subseteq\boldsymbol{S}$;如果 $r_{ij}=q_{ij}$,$i=1,2,\cdots,m$,$j=1,2,\cdots,n$,则称 $\boldsymbol{R}$ 被模糊矩阵 $\boldsymbol{S}$ 相等。

必须指出,一般 $\boldsymbol{R}\cup\boldsymbol{R}^C\neq\boldsymbol{F}$,$\boldsymbol{R}\cup\boldsymbol{R}^C\neq\boldsymbol{O}$,即对模糊矩阵互补律不成立。其中,$\boldsymbol{O}$、$\boldsymbol{F}$ 分别称为零矩阵及全矩阵,即

$$\boldsymbol{O}=\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix},\quad \boldsymbol{F}=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{5-12}$$

与模糊集的 λ-截集相似,在模糊矩阵的矩阵截集定义为

$$\boldsymbol{R}_\lambda=[\lambda r_{ij}]_{m\times n},\quad \lambda\in[0,1] \tag{5-13}$$

或

$$\boldsymbol{R}_\lambda=\{(x,y)\mid\mu_R(x,y)\geqslant\lambda\} \tag{5-14}$$

例 6:身高集合 $X=\{150,155,160,165\}$,体重集合 $Y=\{45,49.5,54,58.5\}$,根据国际上常用的人的体重计算公式:标准体重=[身高(cm)−100]×0.9(kg),则 $X\times Y$ 中的 $\boldsymbol{R}$ 为

$$\boldsymbol{R}=\begin{bmatrix} 1 & 0.75 & 0.3 & 0 \\ 0.75 & 1 & 0.75 & 0.3 \\ 0.3 & 0.75 & 1 & 0.75 \\ 0 & 0.3 & 0.75 & 1 \end{bmatrix}$$

则 $R_{0.75}=\{(x,y)\mid\mu_R(x,y)\geqslant 0.75\}$,即 $R_{0.75}=\{(x_1,y_1),(x_1,y_2),(x_2,y_1),(x_2,y_2),(x_2,y_3),(x_3,y_2),(x_3,y_3),(x_3,y_4),(x_4,y_3),(x_4,y_4)\}$。

如果用矩阵表示,则

$$R_\lambda = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

5.4.2 模糊关系的合成

模糊关系合成是指由第一个集合和第二个集合之间的模糊关系及第二个集合和第三个集合之间的模糊关系得到第一个集合和第三个集合之间的模糊关系的一种运算。

模糊关系的合成的计算方法有取大-取小合成法、取大-乘积合成法、加法-相乘合成法。下面给出常用的取大-取小合成法的定义。

定义 23:设 $\boldsymbol{R}$ 是 $X\times Y$ 中的模糊关系,$\boldsymbol{S}$ 是 $Y\times Z$ 中的模糊关系,$\boldsymbol{R}$ 和 $\boldsymbol{S}$ 的合成是下列定义在 $X\times Z$ 上的模糊关系 $\boldsymbol{Q}$,记作

$$\boldsymbol{Q} = \boldsymbol{R} \circ \boldsymbol{S} \tag{5-15}$$

或

$$\mu_{R\circ S}(x,z) = \vee \{\mu_R(x,y) \wedge \mu_S(y,z)\} \tag{5-16}$$

式中,$\wedge$ 代表取小,$\vee$ 代表取大,"$\circ$"表示合成运算。因此,这一计算方法称为取大-取小(max-min)合成法。

定义 24:设 $\boldsymbol{Q}=(q_{ij})_{n\times m}$,$\boldsymbol{R}=(r_{jk})_{m\times l}$ 是两个模糊矩阵,它们的合成 $\boldsymbol{Q}\circ\boldsymbol{R}$ 指的是一个 n 行 l 列的模糊矩阵 $\boldsymbol{S}$,$\boldsymbol{S}$ 的第 i 行第 k 列的元素 s_{ik} 等于 $\boldsymbol{Q}$ 的第 i 行元素与第 k 列对应元素两两先取较小者,然后在所有的结果中取较大者,即

$$s_{ik} = \bigvee_{i=1}^{m} (q_{ij} \wedge r_{jk}), \quad 1 \leqslant i \leqslant n, 1 \leqslant k \leqslant l \tag{5-17}$$

模糊矩阵 $\boldsymbol{Q}$ 与 $\boldsymbol{R}$ 的合成 $\boldsymbol{Q}\circ\boldsymbol{R}$ 又称为 $\boldsymbol{Q}$ 对 $\boldsymbol{R}$ 的模糊乘积,或者称模糊矩阵的乘法。

例 7:现对某一餐馆的品质进行评判,评判的指标包括饭菜口感、饭菜色相、环境舒适度、服务态度及卫生状况 5 个方面,用论域 Y 来表示,即

$$Y = \{饭菜口感,饭菜色相,环境舒适度,服务态度,卫生状况\}$$

而评判论域用 Z 来表示,即

$$Z = \{很好,较好,可以,不好\}$$

现邀请一些专家对这一餐馆给出评价,即得出 $Y\times Z$ 中的模糊关系 S,表 5-2 列出了 $Y\times Z$ 中的模糊关系 S。

表 5-2 $Y\times Z$ 中的模糊关系 S

Y	Z			
	很好	较好	可以	不好
饭菜口感	0.8	0.15	0.05	0
饭菜色相	0.7	0.2	0.1	0
环境舒适度	0.5	0.3	0.15	0.05
服务态度	0.4	0.25	0.2	0.15
卫生状况	0	0.2	0.3	0.5

表 5-2 使用模糊矩阵可表示为

$$S = \begin{bmatrix} 0.8 & 0.15 & 0.05 & 0 \\ 0.7 & 0.2 & 0.1 & 0 \\ 0.5 & 0.3 & 0.15 & 0.05 \\ 0.4 & 0.25 & 0.2 & 0.15 \\ 0 & 0.2 & 0.3 & 0.5 \end{bmatrix}$$

在对餐馆作综合评定时,各指标对综合评定结果的影响因子不同,对餐馆饭菜口感(0.5)和餐馆卫生状况(0.25)要求较高,其次为服务态度(0.1)和饭菜色相(0.1),对环境的舒适度要求较低(0.05),则得出影响因子集合 X 与评判指标 Y 之间的模糊关系 R,表 5-3 列出了 $X\times Y$ 中的模糊关系 R。

表 5-3 $X\times Y$ 中的模糊关系 R

X	Y				
	饭菜口感	饭菜色相	环境舒适度	服务态度	卫生状况
影响因子	0.5	0.1	0.05	0.1	0.25

表 5-3 使用模糊矩阵可表示为 $\boldsymbol{R}=[0.5 \quad 0.1 \quad 0.05 \quad 0.1 \quad 0.25]$。

现要求在不同权重因子下,做出餐馆综合品质结论。

此时就要求做模糊关系的合成,即餐馆的综合品质 $\boldsymbol{Q}$ 为

$$\boldsymbol{Q} = \boldsymbol{R} \circ \boldsymbol{S} = [0.5 \quad 0.1 \quad 0.05 \quad 0.1 \quad 0.25] \circ \begin{bmatrix} 0.8 & 0.15 & 0.05 & 0 \\ 0.7 & 0.2 & 0.1 & 0 \\ 0.5 & 0.3 & 0.15 & 0.05 \\ 0.4 & 0.25 & 0.2 & 0.15 \\ 0 & 0.2 & 0.3 & 0.5 \end{bmatrix}$$

根据取大-取小的原则

$q_1=(0.5\wedge 0.8)\vee(0.1\wedge 0.7)\vee(0.05\wedge 0.5)\vee(0.1\wedge 0.4)\vee(0.25\wedge 0)=0.5$

$q_2=(0.5\wedge 0.15)\vee(0.1\wedge 0.2)\vee(0.05\wedge 0.3)\vee(0.1\wedge 0.25)\vee(0.25\wedge 0.2)=0.2$

$q_3=(0.5\wedge 0.05)\vee(0.1\wedge 0.1)\vee(0.05\wedge 0.15)\vee(0.1\wedge 0.2)\vee(0.25\wedge 0.3)=0.25$

$q_4=(0.5\wedge 0)\vee(0.1\wedge 0)\vee(0.05\wedge 0.05)\vee(0.1\wedge 0.15)\vee(0.25\wedge 0.5)=0.25$

即 $\boldsymbol{Q}=\boldsymbol{R}\circ\boldsymbol{S}=[0.5 \quad 0.2 \quad 0.25 \quad 0.25]$。

根据计算结果可知,该餐馆综合品质为很好。

根据模糊关系合成的计算方式可知,模糊关系合成不满足交换律。对于例 7 中,$\boldsymbol{R}\circ\boldsymbol{S}$ 有意义,而 $\boldsymbol{S}\circ\boldsymbol{R}$ 没有意义。

设 $\boldsymbol{R}$、$\boldsymbol{S}(\boldsymbol{T})$及 $\boldsymbol{U}$ 分别为 $X\times Y$、$Y\times Z$ 及 $Z\times W$ 中的模糊关系,则有以下 5 个基本性质。

(1) 结合律:如果 $\boldsymbol{S}\subseteq\boldsymbol{T}$,则有 $\boldsymbol{R}\circ\boldsymbol{S}\subseteq\boldsymbol{R}\circ\boldsymbol{T}$ 或 $\boldsymbol{S}\circ\boldsymbol{U}\subseteq\boldsymbol{T}\circ\boldsymbol{U}$。

(2) 并运算上的弱分配律:$\boldsymbol{R}\circ(\boldsymbol{S}\cup\boldsymbol{T})\subseteq(\boldsymbol{R}\circ\boldsymbol{S})\cup(\boldsymbol{S}\circ\boldsymbol{T})$或$(\boldsymbol{S}\cup\boldsymbol{T})\circ\boldsymbol{U}\subseteq(\boldsymbol{S}\circ\boldsymbol{U})\cup(\boldsymbol{T}\circ\boldsymbol{U})$。

(3) 交运算上的弱分配律:$\boldsymbol{R}\circ(\boldsymbol{S}\cap\boldsymbol{T})\subseteq(\boldsymbol{R}\circ\boldsymbol{S})\cap(\boldsymbol{S}\circ\boldsymbol{T})$或$(\boldsymbol{S}\cap\boldsymbol{T})\circ\boldsymbol{U}\subseteq(\boldsymbol{S}\circ\boldsymbol{U})\cap(\boldsymbol{T}\circ\boldsymbol{U})$。

(4) $\boldsymbol{O}\circ\boldsymbol{R}=\boldsymbol{R}\circ\boldsymbol{O}$ 或 $\boldsymbol{I}\circ\boldsymbol{R}=\boldsymbol{R}\circ\boldsymbol{I}=\boldsymbol{R}$($\boldsymbol{O}$ 为零矩阵,$\boldsymbol{I}$ 为单位矩阵)。

(5) 若 $\boldsymbol{R}_1\subseteq\boldsymbol{R}_2$,$\boldsymbol{S}_1\subseteq\boldsymbol{S}_2$,则 $\boldsymbol{R}_1\circ\boldsymbol{S}_1\subseteq\boldsymbol{R}_2\circ\boldsymbol{S}_2$。

5.4.3 模糊关系的性质

定义 25：设 $\boldsymbol{R}$ 是 X 中的模糊关系。若对 $\forall x\in X$，都有 $\mu_R(x,x)=1$，则称 R 为具有自反性的模糊关系。

对应于自反关系的模糊矩阵的对角元素为 1。

定义 26：设 $\boldsymbol{R}\in \boldsymbol{U}(X\times X)$，$\boldsymbol{R}^{\mathrm{T}}$ 是 $\boldsymbol{R}$ 的转置。即 $\boldsymbol{R}^{\mathrm{T}}\in U(X\times X)$，并且满足 $\mu_R^{\mathrm{T}}(y,x)\in \mu_R(y,x)$，其中 $(x,y)\in Y\times X$。

关系的转置有以下性质：

- $(\boldsymbol{R}^{\mathrm{T}})^{\mathrm{T}}=\boldsymbol{R}$
- $(\boldsymbol{R}\cup \boldsymbol{Q})^{\mathrm{T}}=\boldsymbol{R}^{\mathrm{T}}\cup \boldsymbol{Q}^{\mathrm{T}}$ 或 $(\boldsymbol{R}\cap \boldsymbol{Q})^{\mathrm{T}}=\boldsymbol{R}^{\mathrm{T}}\cap \boldsymbol{Q}^{\mathrm{T}}$
- $(\boldsymbol{R}\circ \boldsymbol{Q})^{\mathrm{T}}=\boldsymbol{Q}^{\mathrm{T}}\circ \boldsymbol{R}^{\mathrm{T}}$ 或 $(\boldsymbol{R}^n)^{\mathrm{T}}=(\boldsymbol{Q}^{\mathrm{T}})^n$
- $(\boldsymbol{R}^{\mathrm{T}})_\lambda=(\boldsymbol{R}_\lambda)^{\mathrm{T}}$

定义 27：设 $\boldsymbol{R}\in \boldsymbol{U}(X\times X)$，若 $\boldsymbol{R}^{\mathrm{T}}=\boldsymbol{R}$，则称 $\boldsymbol{R}$ 为对称的模糊关系。在有限论域中，称为对称模糊矩阵。

例 8：设身高集合 $X=\{150,155,160,165\}$，体重集合 $Y=\{45,49.5,54,58.5\}$，根据国际上常用的人的体重计算公式：标准体重＝[身高(cm)－100]×0.9(kg)，则对应身高“非标准”体重时，可采用模糊矩阵表示身高、体重与标准体重之间的关系。

$$\boldsymbol{R}=\begin{bmatrix}1 & 0.75 & 0.3 & 0\\ 0.75 & 1 & 0.75 & 0.3\\ 0.3 & 0.75 & 1 & 0.75\\ 0 & 0.3 & 0.75 & 1\end{bmatrix}$$

由于 $\mu_R(1,1)=\mu_R(2,2)=\mu_R(3,3)=\mu_R(4,4)=1$，则 $\boldsymbol{R}$ 为具有自反性的模糊关系；由于 $\boldsymbol{R}^{\mathrm{T}}=\boldsymbol{R}$，则 $\boldsymbol{R}$ 为具有对称性的模糊关系，即 $\boldsymbol{R}$ 是自反的对称模糊矩阵。

定义 28：设 $\boldsymbol{R}\in \boldsymbol{U}(X\times X)$，即 $\boldsymbol{R}$ 是 X 中的模糊关系。若 $\boldsymbol{R}$ 满足 $\boldsymbol{R}\circ\boldsymbol{R}\subseteq\boldsymbol{R}$，则称 $\boldsymbol{R}$ 为传递的模糊关系。

从定义可见，传递性关系包含着它与它自己的关系合成。对于传递性关系可以等价表示为 $\mu_R(x,y)\geqslant\vee(\mu_R(x,y)\wedge\mu_R(y,z))$，$\forall x,y,z\in X$。

例 9：设 $\boldsymbol{R}=\begin{bmatrix}0.1 & 0.5 & 0.8 & 1\\ 0 & 0.2 & 0.6 & 0.8\\ 0 & 0 & 0.3 & 0.7\\ 0 & 0 & 0 & 0.4\end{bmatrix}$，则

$$\boldsymbol{R}\circ\boldsymbol{R}=\begin{bmatrix}0.1 & 0.5 & 0.8 & 1\\ 0 & 0.2 & 0.6 & 0.8\\ 0 & 0 & 0.3 & 0.7\\ 0 & 0 & 0 & 0.4\end{bmatrix}\circ\begin{bmatrix}0.1 & 0.5 & 0.8 & 1\\ 0 & 0.2 & 0.6 & 0.8\\ 0 & 0 & 0.3 & 0.7\\ 0 & 0 & 0 & 0.4\end{bmatrix}$$

$$=\begin{bmatrix}0.1 & 0.2 & 0.5 & 0.7\\ 0 & 0.2 & 0.3 & 0.6\\ 0 & 0 & 0.3 & 0.4\\ 0 & 0 & 0 & 0.4\end{bmatrix}$$

根据定义，$\boldsymbol{R}\circ\boldsymbol{R}\subseteq\boldsymbol{R}$，则 $\boldsymbol{R}$ 为传递的模糊关系。

定义 29：设 R 是 X 中的模糊关系，若 R 具有自反性和对称性，则 R 称为模糊相似关系。若 R 同时具有自反性、对称性和传递性，则称 R 是模糊等价关系。

利用模糊等价关系对事物进行分类，称为模糊聚类分析。

5.4.4 模糊变换

模糊变换是指给定两个集合之间的一个模糊关系，据此将一个集合上的模糊子集经运算得到另一个集合上的模糊子集的过程。

定义 30：称映射 $F:X\to Y$ 为从 X 到 Y 的模糊变换。模糊变换实现了将 X 中的模糊集变为 Y 上的模糊集，实际上实现了论域的转换。

当 X、Y 均为有限集时，映射 $F:\mu_{1\times m}\to\mu_{1\times n}$ 就是模糊变换。

定义 31：给定一个模糊变换 $F:X\to Y$，若存在 $R\subseteq X\times Y$，使得 $\forall A\in X$，有

$$F(A)=A\circ R\in V \tag{5-18}$$

此处 $\mu_{X\circ R}=\vee(\mu_X(x)\wedge\mu_R(x,y))$，$\forall\, y\in Y$，则为线性模糊变换。

例 10：某一水位控制系统，当前水位的模糊集合 $A=(0.6,0.3,0.1)$，水位与阀门开度的模糊矩阵为

$$\boldsymbol{R}=\begin{bmatrix}0.1 & 0.3 & 0.6\\ 0.2 & 0.5 & 0.3\\ 0.6 & 0.3 & 0.1\end{bmatrix}$$

则在当前水位下，阀门开度为

$$\boldsymbol{Y}=\boldsymbol{A}\circ\boldsymbol{R}=\begin{bmatrix}0.6 & 0.3 & 0.1\end{bmatrix}\circ\begin{bmatrix}0.1 & 0.3 & 0.6\\ 0.2 & 0.5 & 0.3\\ 0.6 & 0.3 & 0.1\end{bmatrix}=\begin{bmatrix}0.2 & 0.3 & 0.6\end{bmatrix}$$

在模糊集合论中还有一个重要的定义，即扩张原理。它是指模糊集合 A 经过映射 f 之后，记为 $f(A)$，而 A 和 $f(A)$ 的相应元素的隶属度保持不变，也就是模糊集合 A 的元素隶属度可以通过映射，无保留地传递到模糊集合 $f(A)$ 的相应元素中。

定义 32：设有映射 $f:X\to Y$，并且 A 是 X 中的模糊集合，记 A 在 f 下的像为 $f(A)$，它是 Y 中的模糊集合，并且具有如下隶属度函数

$$\mu_{f(A)}(y)=\begin{cases}\bigvee\limits_{x\in f^{-1}(y)}(\mu_A(x)), & f^{-1}(y)\neq\varnothing\\ 0, & f^{-1}(y)=\varnothing\end{cases} \tag{5-19}$$

即若 $A=\dfrac{\mu_1}{x_1}+\dfrac{\mu_2}{x_2}+\cdots+\dfrac{\mu_m}{x_m}$，则由映射 f 作用之后有 $f(A)$，$f(A)=\dfrac{\mu_1}{f(x_1)}+\dfrac{\mu_2}{f(x_2)}+\cdots+\dfrac{\mu_m}{f(x_m)}$。

当 f 为一一映射时，$f(A)$ 的隶属度函数公式可简化为

$$\mu_{f(A)}(y)=\begin{cases}(\mu_A(x)), & f^{-1}(y)\neq\varnothing\\ 0, & f^{-1}(y)=\varnothing\end{cases}$$

例 11：设 $A=\frac{0.1}{x_1}+\frac{0.3}{x_2}+\frac{0.4}{x_3}+\frac{0.7}{x_4}+\frac{0.5}{x_5}+\frac{0.2}{x_6}$，$Y=\{y_1,y_2,y_3\}$，映射 $f:X\to Y$，它具有 $f(x_1)=y_1$，$f(x_2)=y_2$，$f(x_3)=y_2$，$f(x_4)=y_2$，$f(x_5)=y_3$ 及 $f(x_6)=y_3$，则 $f(y_1)=\{x_1\}$，$f(y_2)=\{x_2,x_3,x_4\}$，$f(y_3)=\{x_5,x_6\}$。得

$$\mu_{f(A)}(y_1)=\bigvee_{\{x_1\}}(0.1)=0.1$$

$$\mu_{f(A)}(y_2)=\bigvee_{\{x_2,x_2,x_4\}}(0.3,0.4,0.7)=0.7$$

$$\mu_{f(A)}(y_3)=\bigvee_{\{x_5,x_6\}}(0.5,0.2)=0.5$$

即 $f(A)=\frac{0.1}{y_1}+\frac{0.7}{y_2}+\frac{0.5}{y_3}$。

5.5 模糊逻辑及模糊推理

模糊集合是经典集合的真实概括，经典集合是模糊集合的特例。使用隶属度函数定义的模糊集合称为模糊逻辑。模糊集合中的隶属度函数用于鉴定“陈述”为“真”的程度。例如，体温为 104℉的患者隶属于“重感冒患者”集合的程度为 0.65。任一体温的患者隶属于“重感冒患者”集合的程度可用图 5-15 所示的隶属度函数图表示。

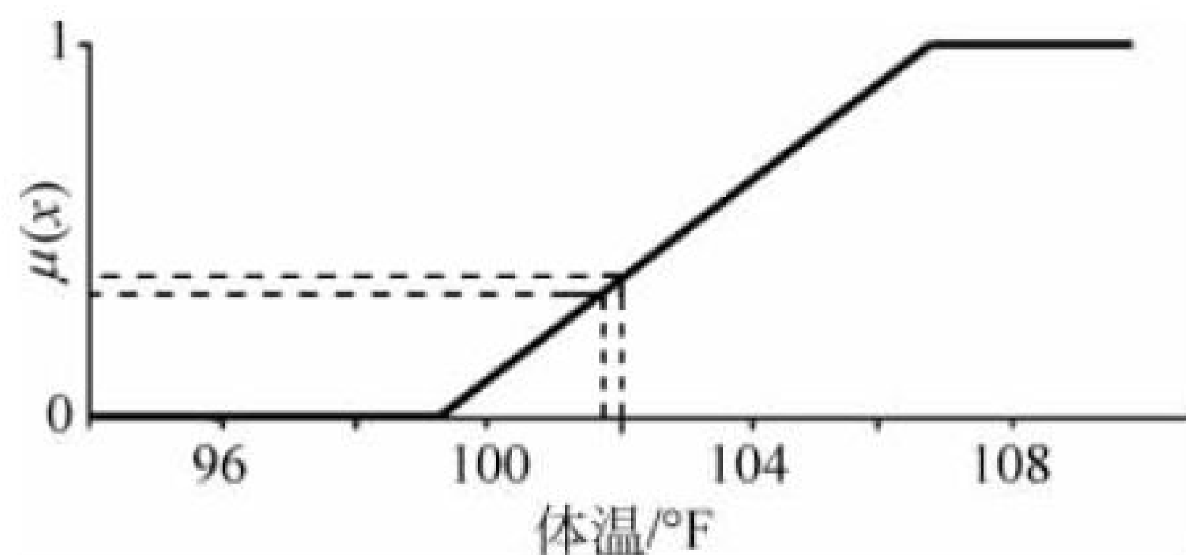


图 5-15 “重感冒”隶属度函数曲线

语言变量是模糊逻辑系统的基本构成，它对同样背景使用多个主观分类进行描述。以发热为例，将描述发热的程度用高烧(strong-fever)、发热(raised)、正常(normal)和低烧(low)4 个语言变量来描述。图 5-16 展示了所有语言变量就“发热”事件的隶属度函数。

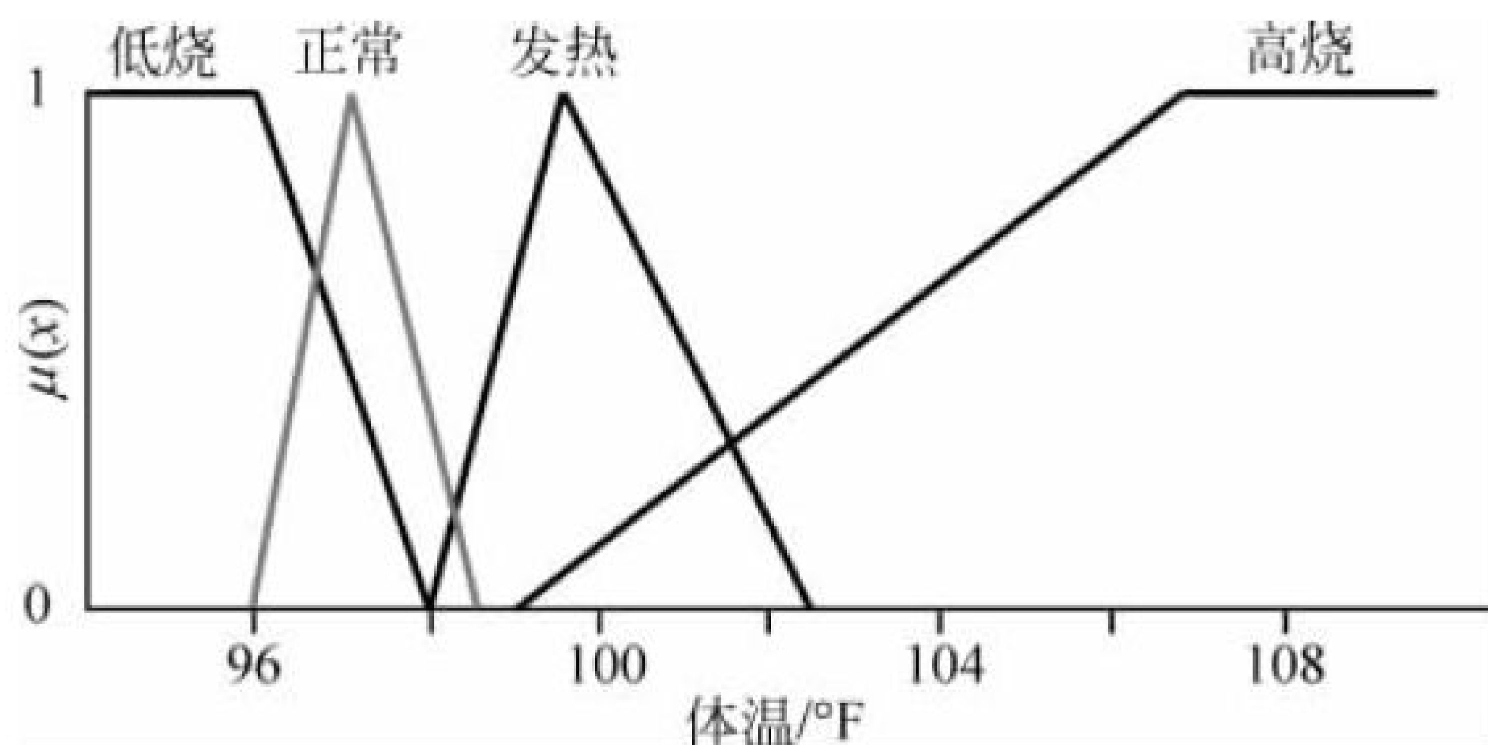


图 5-16 所有语言变量就“发热”事件的隶属度函数曲线

使用模糊隶属度函数后，以华氏温度测量的体温可以转换为语言描述。例如，体温为100℉的患者将可确诊为基本属于发烧状态，并有轻微的高烧现象。

5.5.1 模糊逻辑技术

随着人们对模糊逻辑理解的加深，使用模糊集合的方法不断更新。本书只涉及基于规则的模糊逻辑技术。几乎所有的近期的模糊逻辑应用都是基于该方法。这里简要介绍集装箱起重机控制实例的基于规则的模糊逻辑系统的基本技术。

集装箱起重机控制系统界面如图 5-17 所示。

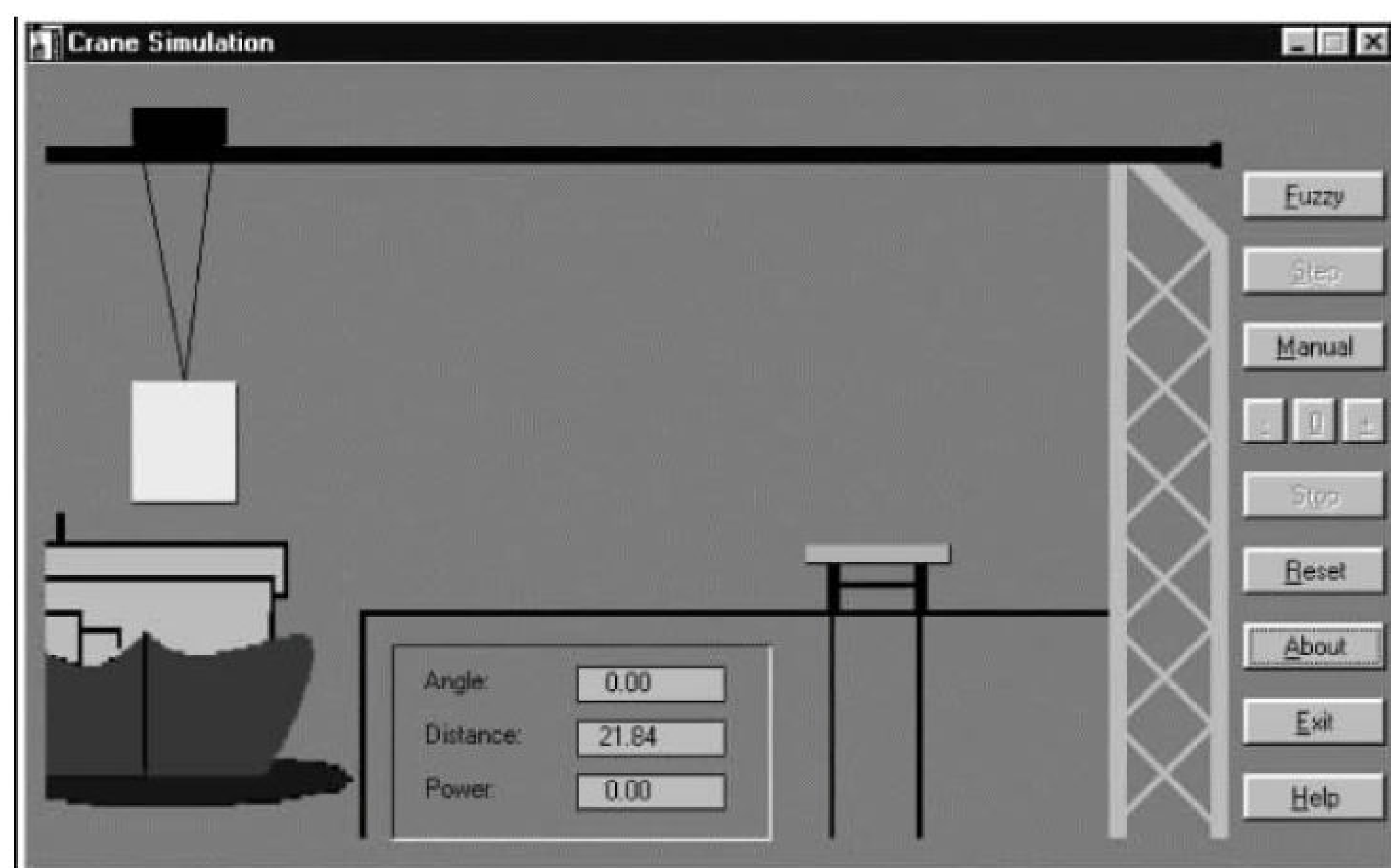


图 5-17 集装箱起重机控制系统

集装箱起重机用于装载集装箱到船上或从港口的船上卸载集装箱。使用连接到起重机头上的软电缆吊起单个集装箱，起重机头采用水平移动方式。当一个集装箱被提起，起重机头开始移动，此时集装箱随着起重机头的移动和惯性力的作用开始晃动。在运输过程中，集装箱的晃动基本不会影响运输过程，但晃动的集装箱必须稳定后才能放下。

解决这个问题有两种方法。第一种方法是准确定位起重机头到目标位置上方，接着一直等待集装箱的摆动达到规定的稳定状态。当然，摆动的集装箱最终会稳定，但等待会造成相当多时间的浪费。由于成本原因一个集装箱船需要在最短时间内被装载和卸载，因此第一种方法不符合成本最小化的要求。第二种方法就是提起集装箱，然后慢慢移动它，致使集装箱不发生晃动，但这一方法也将花费大量的时间。

一个比较折中的方法就是，在操作过程中使用附加电缆固定集装箱的位置来构建集装箱起重机。但这个方案花费较高，很少有起重机利用这个技术。

由于这些原因，大多数集装箱起重机在操作员的指导下对起重机电动机采用连续速度控制。操作员需要控制晃动的同时，确保集装箱在最短时间内达到目标位置。实现这个目标，对于操作人员来说非常不容易，但熟练的操作员能够做到。为了降低操作的难度，工程师们曾尝试采用控制策略实现自动控制，如线性 PID 控制、基于模型控制和模糊逻辑控制。

传统的 PID(比例-积分-微分)控制试验未能成功，因为控制任务是非线性的。当集装箱接近目标时，晃动最小化是重要的；在基于模型的控制试验时，工程师推导出描述起重机

机械行为的数学模型为五阶微分方程式，这在理论上说明基于模型的控制策略是可行的，但试验却也不成功。造成这一策略不成功的原因如下：

(1) 起重机电动机行为不是模型中假设的那样线性。

(2) 起重机头移动时有摩擦。

(3) 在模型中未包含干扰量，如风的干扰。

鉴于以上控制策略的不足，引入了模糊逻辑的语言控制策略。

5.5.2 语言控制策略

在人为控制中，操作员并非按照微分方程式进行控制，甚至无须使用基于模型的控制策略中的电缆长度传感器。一旦操作员提起集装箱，首先使用中功率电动机，以便查看集装箱如何晃动。然后依据晃动的程度，调整电动机功率使集装箱在起重机头后面一点，这时系统将获得最大的传输速度，并且使集装箱的晃动最小。

当接近目标位置时，操作员减小电动机功率或使用负电压刹车。当起重机很接近目标且电压进一步减小或反向时，使集装箱的位置稍微超过起重机头，直到集装箱几乎达到目标位置。最终，电动机功率增加以致起重机头超过目标位置且摆动为0。在整个操作过程中，不需要微分方程式，系统干扰或非线性通过操作员对集装箱位置的观察，依据经验进行补偿。

在对操作员操作过程的分析中，使用了一些经验规则描述控制策略。

(1) 在起动时使用中功率电动机，以便观察集装箱的晃动情况。

(2) 如果已经起动且仍然远离目标，增加电动机功率，以使集装箱到达起重机头后面一点。

(3) 如果接近目标，减小速度以致集装箱在起重机头前面一点。

(4) 当集装箱超过目标，并且晃动为0时，停止电动机。

用距离传感器测量起重机头的位置，用相角传感器测量集装箱晃动相角，并将测量结果应用到自动控制起重机中。使用测量结果描述起重机的当前状况，并采用"如果——则"格式描述经验控制规则。

(1) 如果起重机头与目标位置之间的距离较远，并且集装箱与垂直方向的相角等于0，则使用中功率电动机。

(2) 如果起重机头与目标位置之间的距离较远，并且集装箱与垂直方向的相角小于0，则使用大功率电动机。

(3) 如果起重机头与目标位置之间的距离较近，并且集装箱与垂直方向的相角小于0，则使用中功率电动机。

(4) 如果起重机头与目标位置之间的距离适中，并且集装箱与垂直方向的相角小于0，则使用中功率电动机。

(5) 如果起重机头到达目标位置，并且集装箱与垂直方向的相角等于0，则停止电动机。

从经验控制规则可知，采用"如果——则"格式描述经验控制规则的通用式为

如果＜状态＞则＜动作＞

就集装箱起重机而言，状态由两个条件确定：第一个条件描述起重机头与目标位置之

间的距离值；第二个条件描述集装箱与垂直方向的相角。当两个条件同时满足相应的状态时，系统给出控制策略。

在设置规则时，使用到语言变量。

5.5.3 模糊语言变量

带有模糊性的语言称为模糊语言，如高、矮、胖、瘦、轻、重、缓、急等。此外，在自然语言中有一些词可以表达语气的肯定程度，如“非常”“狠”“极”等；也有一类词，如“大概”“近似于”等，将这些词置于某个词前面，如年轻，则使该词意义变为模糊，如很年轻；还有些词，如“偏向”“倾向于”等可使词义由模糊变为肯定，如倾向于短等。在模糊控制中，常见的模糊语言还有正大、正中、正小、零、负小、负中、负大等。

人类自然语言具有模糊性，而通常的计算机语言有严格的语法规则和语义，不存在任何的模糊性和歧义，即计算机对模糊性缺乏识别和判断能力。为了实现用自然语言跟计算机进行直接对话，就必须把人类的语言和思维过程提炼成数学模型。

语言变量是指以自然或人工语言的词、词组或句子作为值的变量。如模糊控制中的“偏差”“偏差变化率”等，并且语言变量的取值通常不是数，而是用模糊语言表示的模糊集合，如“偏差很大”“偏差大”“偏差适中”“偏差小”和“偏差较小”。

定义 33：一个语言变量可定义为多元组$(x,T(x),U,G,M)$。其中，x为变量名；$T(x)$为x的词集，即语言值名称的集合；U为论域；G是产生语言值名称的语法规则；M是与各语言值含义有关的语法规则。语言变量的每个语言值对应一个定义在论域U中的模糊数。语言变量基本词集把模糊概念与精确值联系起来，实现对定性概念的定量化及定量数据的定性模糊化。

依然以偏差为例，则T(偏差)={很大、大、适中、小、较小}。

上述每个模糊语言(如大、适中等)是定义在论域U上的一个模糊集合。设论域$U=[0,5]$，则可大致认为小于1为小，2左右为适中，大于3以上为大。

语法规则是根据原子单词来生成的语言值集合$T(x)$中各个合成词语的语法规则。

(1) 前缀限制词H方式，在原子单词C之前引入算子H概念，形成合成语言词$T=HC$。例如，“极”“很”“相当”等都可以作为算子来处理。算子有很多种，经常使用的有语气算子(“极”“很”)、散漫化算子(“略”“微”)、概率算子(“大概”“将近”)、判定化算子(“倾向于”“多半是”)等。

(2) 加连接词“或”“且”和否定词“非”，如“非大于”等。

(3) 混合式，即上述两种合成方式重复或交叉使用，形成各种复杂的语言值。

以偏差为例的语言变量的结构图如图5-18所示。

5.5.4 模糊命题与模糊条件语句

人们把具有模糊概念的陈述句称为模糊命题，如“天气很热”。模糊命题的标志符通常用大写字母P、Q、R等下面加波浪～表示。

表征模糊命题真实程度的量叫作模糊命题的真值，记作

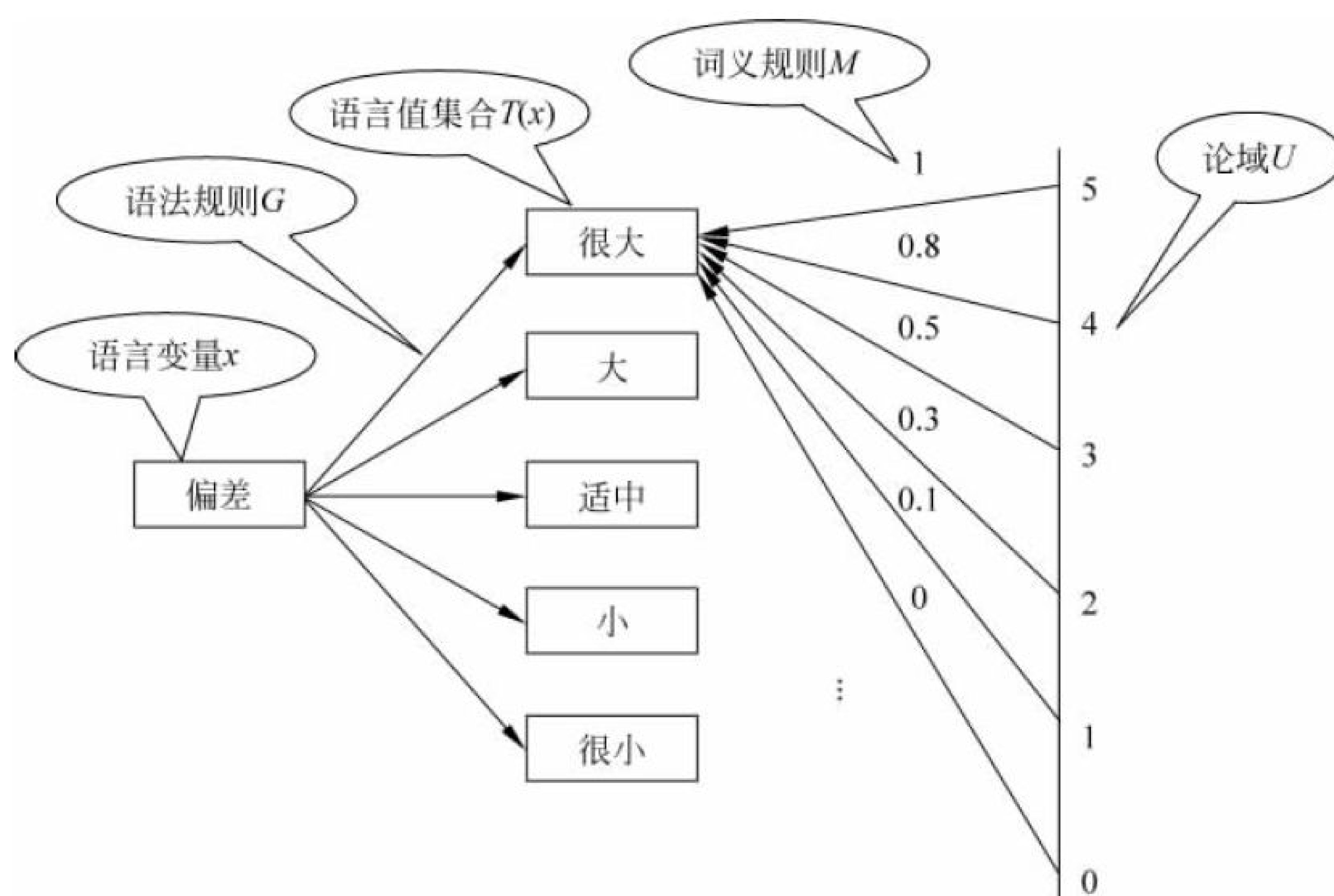


图 5-18　以偏差为例的语言变量的结构图

$$V(\underset{\sim}{P}) = x, \quad 0 \leqslant x \leqslant 1 \tag{5-20}$$

当 $V(\underset{\sim}{P})=1$ 时，表示 $\underset{\sim}{P}$ 陈述的信息完全真；当 $V(\underset{\sim}{P})=0$ 时，表示 $\underset{\sim}{P}$ 陈述的信息完全假；而当 $V(\underset{\sim}{P})$ 的值介于 0～1 时，表示 $\underset{\sim}{P}$ 陈述的信息不完全真，也不完全假，并且 $V(\underset{\sim}{P})$ 的值越接近于 1，表示 $\underset{\sim}{P}$ 陈述的信息越真实。模糊命题比二值逻辑中的命题更能符合人脑的思维方式，反映了真或假的程度。

模糊命题的一般形式为 $\underset{\sim}{P}$："u 是 A"(或 u is A)。其中，u 是个体变元，它属于论域 U，即 $u \in U$；A 是某个模糊概念所对应的模糊集合。模糊命题的真值由该变元对模糊集合的隶属程度表示，定义为

$$V(\underset{\sim}{P}) = \mu_A(u) \tag{5-21}$$

在模糊命题中，"is A"部分是表示一个个体模糊性质或多个个体之间的模糊关系的部分，称为模糊谓词。和二值逻辑一样，使用析取、合取、取非、蕴涵及等价运算可构成复合模糊命题。

设有模糊命题 $\underset{\sim}{P}$：偏差大；$\underset{\sim}{Q}$：偏差变化率小，则：

- 析取：表示两者间的关系为"或"，记为 $\underset{\sim}{P} \cup \underset{\sim}{Q}$，其真值为 $V(\underset{\sim}{P}) \vee V(\underset{\sim}{Q}) = \mu_A(p) \vee \mu_B(q)$，意为偏差大或偏差变化率小，即两者满足其一即可。
- 合取：表示两者间的关系为"且"，记为 $\underset{\sim}{P} \cap \underset{\sim}{Q}$，其真值为 $V(\underset{\sim}{P}) \wedge V(\underset{\sim}{Q}) = \mu_A(p) \wedge \mu_B(q)$，意为偏差大，并且偏差变化率小，即要求两者同时满足。
- 取非：表示两者间的关系为"非"，记为 $\underset{\sim}{P}^C$，其真值为 $1-V(\underset{\sim}{P})=1-\mu_A(p)$，意为偏差不大。
- 蕴涵：表示两者间的关系为"若……则……"，记为 $V(\underset{\sim}{P}) \rightarrow V(\underset{\sim}{Q})$。
- 等价：表示两者间的关系为"互相蕴含"，记为 $V(\underset{\sim}{P}) \leftrightarrow V(\underset{\sim}{Q})$。

从真值表达式可知,模糊命题真值之间的运算,也就是其相应隶属度函数之间的运算。

在使用模糊策略时,会用到一系列模糊控制规则,如"如果起重机头与目标位置之间的距离较远,并且集装箱与垂直方向的相角等于0,则使用中等功率电动机",或者"如果偏差较大,而偏差变化率较小,则阀门半开"等。其中"远""中等""大""小""半开"等词均为模糊词,这些带模糊词的条件语句就是模糊条件语句。

在模糊控制中,经常用到3种条件语句。

(1) If 条件 then 语句,其简记形式为 if A then B。其中 if A 部分称为前件或条件部分,then B 部分称为后件或结论部分。

例句:如果水位达到要求,则关闭进水阀门。

(2) If 条件 then 语句 1 else 语句 2,其简记形式为 if A then B else C。

例句:如果苹果比橘子贵,则买橘子;否则,买苹果。

(3) If 条件 1 and 条件 2 then 语句,其简记形式为 if A and B then C。

例句:如果他跑得快,并且球技好,则让他当前锋。

5.5.5 判断与推理

判断和推理是思维形式的一种。判断是概念与概念的联合,而推理则是判断与判断的联合。推理根据一定的原则,从一个或几个已知判断引出一个新判断。一般情况下,推理包含两个部分的判断:一部分是已知的判断,作为推理的出发点,叫作前提或前件;由前提所推出的新判断,叫做结论或后件。

只有一个前提的推理称为直接推理,由两个或两个以上前提的推理称为间接推理。间接推理依据认识的方向,又可分为演绎推理、归纳推理和类比推理等。

演绎推理是前提与结论之间有蕴涵关系的推理。演绎推理中最常用的形式是假言推理,有肯定式推理和否定式推理两类。

肯定式:

大前提(规则)	若 x 是 A,则 y 是 B
小前提(已知)	x 是 A
结论	y 是 B

否定式:

大前提(规则)	若 x 是 A,则 y 是 B
小前提(已知)	y 不是 B
结论	x 不是 A

以上就是"三段论"推理模式,用数学形式表达如下:

$A \to B$	$A \to B$
A	B^c
B	A^c
肯定式	否定式

"三段论"给出了在大前提 $A \to B$ 之下,若小前提是 A,则可推出结论为 B。然而,当小

前提不是严格的 A，而是在某种程度上接近于 A，记为 A'，此时结论应该是什么呢？“三段论”没能给出答案，即三段论对模糊性问题的推理无能为力，此时需要使用模糊推理方法。

5.5.6 模糊推理

模糊推理又称模糊逻辑推理，是应用模糊关系来表示模糊条件句，将推理的判断过程转化为对隶属度的合成及演算过程。即已知模糊命题(包括大前提和小前提)，推出新的模糊命题作为结论的过程。模糊推理即近似推理，这两个术语不加区分，可以混用。

L. A. Zadeh 在 1973 年对于模糊命题“若 A 则 B”，利用模糊关系的合成运算提出了一种近似推理的方法，称为“关系合成推理法”，简称 CRI 法，是实际控制中应用较广的一种模糊推理算法。其原理表述为：用一个模糊集合表述大前提中全部模糊条件语句前件的基础变量和后件的基础变量间的关系；用一个模糊集合表述小前提；进而用基于模糊关系的模糊变换运算给出推理结果。

常用的推理方法有 Zadeh 的推理方法、Mamdani 推理方法、多输入模糊推理和多输入多规则推理。

1. Zadeh 的推理方法

设 A 是 X 上的模糊集合，B 是 Y 上的模糊集合，模糊蕴涵关系“若 A 则 B”，用 $A\rightarrow B$ 表示。Zadeh 把它定义成 $X\times Y$ 的模糊关系，即

$$R = A \rightarrow B = (A\times B)\cup(A^c\times Y) \tag{5-22}$$

其隶属度函数式为 $R(x,y)=[A(x)\wedge B(x)]\vee(1-A(x))$。

给定一个模糊关系 R，就决定了一个模糊变换，利用模糊关系的合成有如下推理规则。

(1) 已知模糊蕴涵关系 $A\rightarrow B$ 的模糊关系 R，对于给定的 A'，$A'\in X$，则可推出结论 B'，$B'\in Y$，$B'=A'\circ R$。即当 Y 为有限论域时，

$$B'(y) = \vee\{A'(x)\wedge[A(x)\wedge B(y)\vee(1-A(x))]\} \tag{5-23}$$

(2) 已知模糊蕴涵关系 $A\rightarrow B$ 的模糊关系 R，对于给定的 B'，$B'\in Y$，则可推出结论 A'，$A'\in X$，$A'=R\circ B'$。即当 X 为有限论域时，

$$A'(x) = \vee\{[A(x)\wedge B(y)\vee(1-A(x))]\wedge B'(y)\} \tag{5-24}$$

2. Mamdani 推理方法

Mamdani 的推理方法本质上是一种 CRI 法，只是 Mamdani 把模糊蕴涵关系 $A\rightarrow B$ 用 A 和 B 的笛卡儿积表示，即 $R=A\rightarrow B=A\times B$，也可写为 $R(x,y)=A(x)\times B(y)$。

已知模糊蕴涵关系 $A\rightarrow B$ 的模糊关系 R，对于给定的 A'，$A'\in X$，则可推出结论 B'，$B'\in Y$，$B'=A'\circ R$。即当 Y 为有限论域时，

$$B'(y) = \vee\{A'(x)\wedge[A(x)\wedge B(y)]\} \tag{5-25}$$

或者使用隶属度函数表示为

$$\begin{aligned}\mu_{B'}(y) &= \vee\{\mu_{A'}(x)\wedge[\mu_A(x)\wedge\mu_B(y)]\}\\ &= \vee\{\mu_{A'}(x)\wedge\mu_A(x)\}\wedge\mu_B(y)\\ &= \alpha\wedge\mu_B(y)\end{aligned} \tag{5-26}$$

其中，$\alpha=\vee\{\mu_{A'}(x)\wedge\mu_A(x)\}$，是模糊集 A'与 A 交集的高度，如图 5-19 所示。也可表示为 $\alpha=H(A'\cap A)$，α 可以看成是 A'对 A 的适配程度。

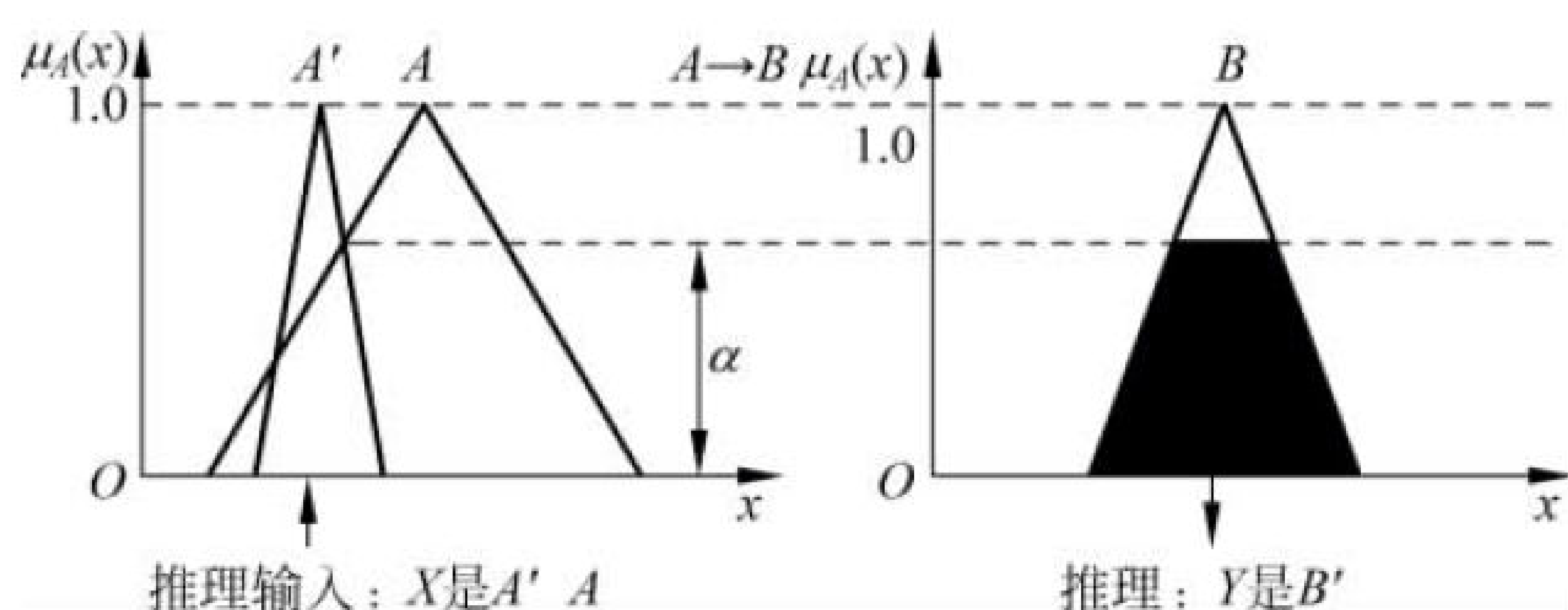


图 5-19 $\alpha=\vee\{\mu_{A'}(x)\wedge\mu_A(x)\}$的图示

根据 Mamdani 推理方法，结论可用此适配度与模糊集合进行模糊与，即取小运算（min）而得到。在图形上就是用基准去切割 B，便可得到推论结果，所以这种方法经常又形象地被称为削顶法。

已知模糊蕴涵关系 $A\rightarrow B$ 的模糊关系 R，对于给定的 B'，$B'\in Y$，则可推出结论 A'，$A'\in X$，$A'=R\circ B'$，其中"$\circ$"表示合成运算。即当 X 为有限论域时，$A'(x)=\vee\{[A(x)\wedge B(y)]\wedge B'(y)\}$，或者使用隶属度函数表示为

$$\mu_{A'}=\vee\{[\mu_A(x)\wedge B(y)]\wedge B'(y)\} \tag{5-27}$$

例 12：设 $A\in X$，$B\in Y$，$A=$数量多，$B=$质量大。论域 X(数量)$=\{0,2,4,6,8,10\}$，$\mu_A(x)=\frac{0}{0}+\frac{0.1}{2}+\frac{0.3}{4}+\frac{0.6}{6}+\frac{0.9}{8}+\frac{1}{10}$，$Y$(质量)$=\{0,1,2,3,4,5,6,7\}$，$\mu_B(y)=\frac{0}{0}+\frac{0.1}{1}+\frac{0.2}{2}+\frac{0.4}{3}+\frac{0.6}{4}+\frac{0.8}{5}+\frac{0.9}{6}+\frac{1}{7}$。"若 A 则 B"（若数量多，则质量大）为推论的大前提，给出模糊关系 $R=A\rightarrow B$。使用 Mamdani 推理方法推导出给定 A'，$\mu_{A'}(x)=\frac{0}{0}+\frac{0.2}{2}+\frac{0.5}{4}+\frac{0.8}{6}+\frac{1}{8}+\frac{0.8}{10}$，在"数量较多"情况下的结论 B'（"质量较大"）。

由式 $\alpha=\vee\{\mu_{A'}(x)\wedge\mu_A(x)\}$，先求出 A'对 A 的适配度为

$$\begin{aligned}\alpha&=\vee\left\{\frac{0\wedge 0}{0}+\frac{0.1\wedge 0.2}{2}+\frac{0.3\wedge 0.5}{4}+\frac{0.6\wedge 0.8}{6}+\frac{0.9\wedge 1}{8}+\frac{1\wedge 0.8}{10}\right\}\\&=\vee\left\{\frac{0}{0}+\frac{0.1}{2}+\frac{0.3}{4}+\frac{0.6}{6}+\frac{0.9}{8}+\frac{0.8}{10}\right\}\\&=0.9\end{aligned}$$

然后用 α 切割 B 的隶属度函数：

$$\begin{aligned}\mu_{B'}(y)&=\alpha\wedge\mu_B(y)\\&=0.9\wedge\left(\frac{0}{0}+\frac{0.1}{1}+\frac{0.2}{2}+\frac{0.4}{3}+\frac{0.6}{4}+\frac{0.8}{5}+\frac{0.9}{6}+\frac{1}{7}\right)\\&=0.9\wedge\left(\frac{0}{0}+\frac{0.1}{1}+\frac{0.2}{2}+\frac{0.4}{3}+\frac{0.6}{4}+\frac{0.8}{5}+\frac{0.9}{6}+\frac{1}{7}\right)\\&=\frac{0}{0}+\frac{0.1}{1}+\frac{0.2}{2}+\frac{0.4}{3}+\frac{0.6}{4}+\frac{0.8}{5}+\frac{0.9}{6}+\frac{0.9}{7}\end{aligned}$$

3. 多输入模糊推理方法

已知推理大前提的条件为"if A and B then C"，$A\in F(x)$，$B\in F(y)$，$C\in F(z)$，模糊蕴涵关系为

$$R = A \times B \times C = (A \times B) \rightarrow C \tag{5-28}$$

或

$$R(x,y,z) = A(x) \wedge B(y) \wedge C(z)$$

当已知输入 A'、B'，小前提为 A'且 B'，则可推出 C'

$$C' = (A' \times B') \circ R \tag{5-29}$$

其中，$A'=F(x)$，$B'=F(y)$，$C'=F(z)$。即

$$C' = (A' \times B') \circ [(A \times B) \rightarrow C] \tag{5-30}$$

例 13：以模糊自动洗衣机为例，已知泥污量适中为 $\boldsymbol{A}$，$\mu_A(x)=\frac{0.3}{1}+\frac{1}{2}+\frac{0.5}{4}$，油脂量适中为 $\boldsymbol{B}$，$\mu_B(y)=\frac{0.2}{1}+\frac{1}{2}+\frac{0.7}{3}$，洗涤时间适中为 $\boldsymbol{C}$，$\mu_C(z)=\frac{0.3}{3}+\frac{1}{6}+\frac{0.7}{9}$。

已知泥污量多为 $\boldsymbol{A}'$，$\mu_{A'}(x)=\frac{0.1}{1}+\frac{0.6}{2}+\frac{0.9}{3}$，油脂量多为 $\boldsymbol{B}'$，$\mu_{B'}(y)=\frac{0.1}{1}+\frac{0.7}{2}+\frac{1}{3}$，求泥污量大且油脂量多的情况下，洗涤时间长 $\boldsymbol{C}'$。

由于 $\boldsymbol{R}=\boldsymbol{A}\times\boldsymbol{B}\times\boldsymbol{C}=(\boldsymbol{A}\times\boldsymbol{B})\rightarrow\boldsymbol{C}$，则

$$\boldsymbol{R}_1 = \boldsymbol{A} \times \boldsymbol{B} = \begin{bmatrix} 0.3 \\ 1 \\ 0.5 \end{bmatrix} \cdot [0.2 \quad 1 \quad 0.7] = \begin{bmatrix} 0.2 & 0.3 & 0.3 \\ 0.2 & 1 & 0.7 \\ 0.2 & 0.5 & 0.5 \end{bmatrix}$$

把 $\boldsymbol{R}_1$ 写成列向量的形式，即

$$\boldsymbol{R}_1^{\mathrm{T}} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.3 \\ 0.2 \\ 1 \\ 0.7 \\ 0.2 \\ 0.5 \\ 0.5 \end{bmatrix}$$

$$\boldsymbol{R} = \boldsymbol{R}_1^{\mathrm{T}} \times \boldsymbol{C} = \boldsymbol{R}_1^{\mathrm{T}} \times [0.3 \quad 1 \quad 0.7] = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.3 \\ 0.2 \\ 1 \\ 0.7 \\ 0.2 \\ 0.5 \\ 0.5 \end{bmatrix} \cdot [0.3 \quad 1 \quad 0.7] = \begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \\ 0.2 & 0.2 & 0.2 \\ 0.3 & 1 & 0.7 \\ 0.3 & 0.7 & 0.7 \\ 0.2 & 0.2 & 0.2 \\ 0.3 & 0.5 & 0.5 \\ 0.3 & 0.5 & 0.5 \end{bmatrix}$$

由于 $\boldsymbol{C}'=(\boldsymbol{A}'\times\boldsymbol{B}')\circ\boldsymbol{R}$，令 $\boldsymbol{R}_2=\boldsymbol{A}'\times\boldsymbol{B}'$，则

$$\boldsymbol{R}_2=\begin{bmatrix}0.1\\0.6\\0.9\end{bmatrix}\cdot\begin{bmatrix}0.1 & 0.7 & 1\end{bmatrix}=\begin{bmatrix}0.1 & 0.1 & 0.1\\0.1 & 0.6 & 0.6\\0.1 & 0.7 & 0.9\end{bmatrix}$$

把 $\boldsymbol{R}_2$ 写成行向量的形式，即

$$\boldsymbol{R}_2^{\mathrm{T}}=\begin{bmatrix}0.1 & 0.1 & 0.1 & 0.1 & 0.6 & 0.6 & 0.1 & 0.7 & 0.9\end{bmatrix}$$

则

$$\begin{aligned}\boldsymbol{C}'&=(\boldsymbol{A}'\times\boldsymbol{B}')\circ\boldsymbol{R}=\boldsymbol{R}_2^{\mathrm{T}}\\&=\begin{bmatrix}0.1 & 0.1 & 0.1 & 0.1 & 0.6 & 0.6 & 0.1 & 0.7 & 0.9\end{bmatrix}\circ\begin{bmatrix}0.2 & 0.2 & 0.2\\0.3 & 0.3 & 0.3\\0.3 & 0.3 & 0.3\\0.2 & 0.2 & 0.2\\0.3 & 1 & 0.7\\0.3 & 0.7 & 0.7\\0.2 & 0.2 & 0.2\\0.3 & 0.5 & 0.5\\0.3 & 0.5 & 0.5\end{bmatrix}\\&=\begin{bmatrix}0.3 & 0.6 & 0.7\end{bmatrix}\end{aligned}$$

或

$$\boldsymbol{C}'=\frac{0.3}{3}+\frac{0.6}{6}+\frac{0.7}{9}$$

用图形方式来说明两输入推理法过程，则有：

大前提（规则）	若 A 且 B，则 C
小前提（已知）	若 A' 且 B'
结论	$C'=(A'\times B')\circ[(A\times B)\to C]$

其中，$A\in F(x)$，$B\in F(y)$，$C\in F(z)$。

对于多维模糊条件语句 R：if A and B then C，可分解为 R'：if A then C，并且 R''：if B then C，则由 R 作近似推理的结论 C' 等于 R' 和 R'' 的"交"运算，$C'=R'\wedge R''$，即

$$C'=A'\circ(A\times C)\cap B'\circ(B\times C)$$

其隶属度函数为

$$\begin{aligned}\mu_{C'}(z)&=\bigvee_{x\in X}\{\mu_{A'}(x)\wedge[\mu_A(x)\wedge\mu_C(z)]\}\cap\bigvee_{y\in Y}\{\mu_{B'}(y)\wedge[\mu_A(y)\wedge\mu_C(z)]\}\\&=\bigvee_{x\in X}\{\mu_{A'}(x)\wedge\mu_A(x)\}\wedge\mu_C(z)\cap\bigvee_{y\in Y}\{\mu_{B'}(y)\wedge\mu_A(y)\}\wedge\mu_C(z)\\&=(\alpha_A\wedge\mu_C(z))\cap(\alpha_B\wedge\mu_C(z))\\&=(\alpha_A\wedge\alpha_B)\wedge\mu_C(z)\end{aligned}$$

这在 Mamdani 推理削顶法中的几何意义是，像单输入情况一样，分别求出 A' 对 A、B' 对 B 的隶属度 α_A 和 α_B，并取这两个之中小的一个值作为总的模糊推理前件的隶属度，再以此为基准去切割推理后件的隶属度函数，便得到结论 C^*，推理过程如图 5-20 所示。

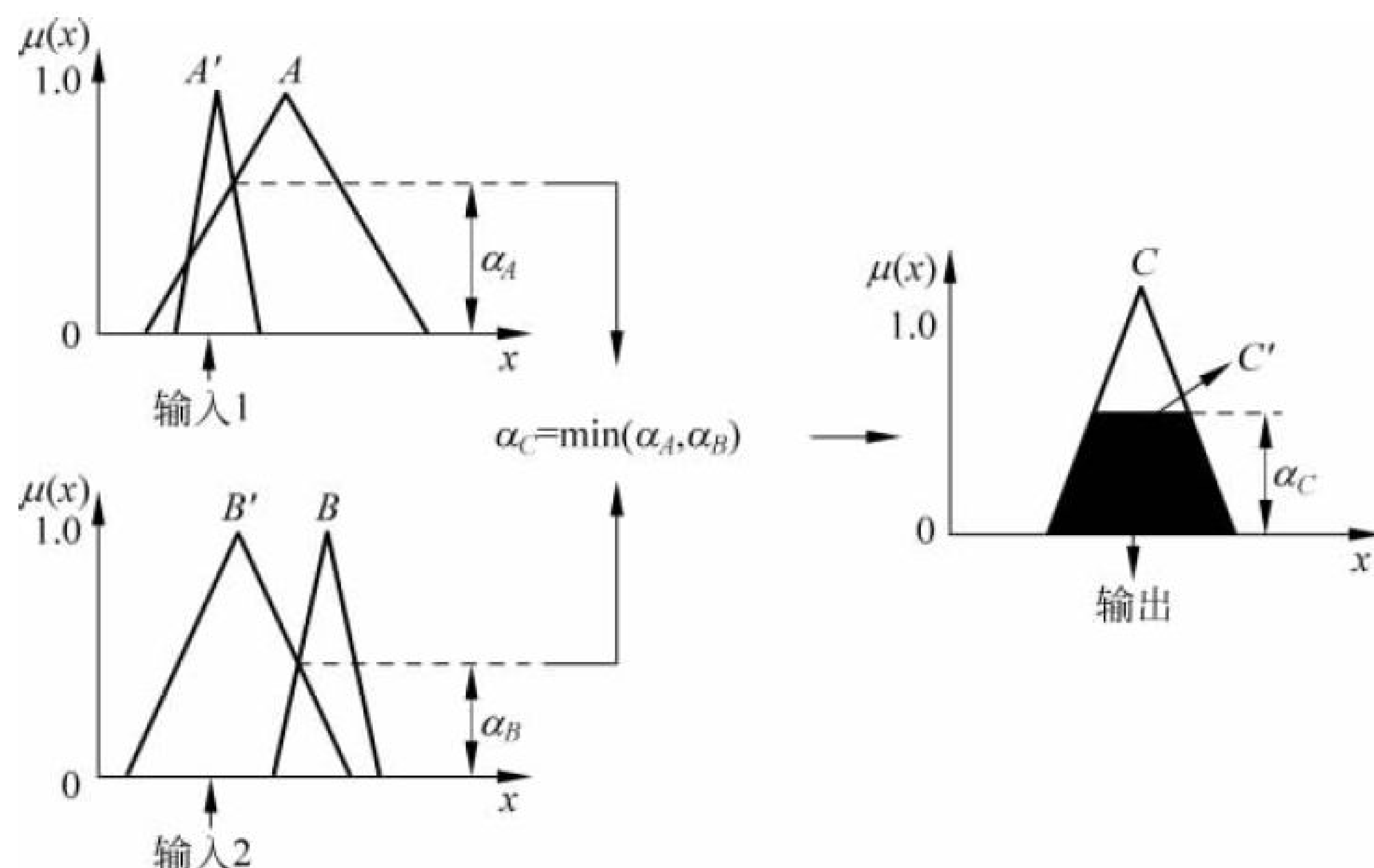


图 5-20 二维输入 Mamdani 推理过程

4. 多输入多规则推理方法

以两输入的多规则为例,其形式为:

	若 A_1 且 B_1,则 C_1,否则
大前提(规则)	若 A_2 且 B_2,则 C_2,否则
	⋮
小前提(已知)	若 A_n 且 B_n,则 C_n,否则
	若 A' 且 B'
结论	C'

其中,A_i 和 A'、B_i 和 B'、C_i 和 C' 分别是不同论域 X、Y、Z 的模糊集合,“否则”表示“或”运算,可写为并集形式

$$C' = (A' \times B') \circ \{[(A_1 \times B_1) \to C_1] \cup \cdots \cup [(A_n \times B_n) \to C_n]\} \tag{5-31}$$

其中,$C_i' = (A' \times B') \circ [(A_i \times B_i) \to C_i] = [A' \circ (A_i \to C_i)] \cap [B' \circ (B_i \to C_i)]$,$i = 1, 2, \cdots, n$。

其隶属度函数为

$$\begin{aligned}
\mu_{C_i'}(z) &= \bigvee_{x \in X} \{\mu_{A'}(x) \wedge [\mu_{A_i}(x) \wedge \mu_{C_i}(z)]\} \cap \bigvee_{y \in Y} \{\mu_{B'}(y) \wedge [\mu_{B_i}(y) \wedge \mu_{C_i}(z)]\} \\
&= \bigvee_{x \in X} [\mu_{A'}(x) \wedge \mu_{A_i}(x)] \wedge \mu_{C_i}(z) \cap \bigvee_{y \in Y} [\mu_{B'}(y) \wedge \mu_{B_i}(y)] \wedge \mu_{C_i}(z) \\
&= (\alpha_{A_i} \wedge \mu_{C_i}(z)) \cap (\alpha_{B_i} \wedge \mu_{C_i}(z)) \\
&= (\alpha_{A_i} \wedge \alpha_{B_i}) \wedge \mu_{C_i}(z) \qquad i = 1, 2, \cdots, n。
\end{aligned} \tag{5-32}$$

如果有两条二维输入规则,则得到两个结论

$$R_1: \mu_{C_1'}(Z) = \alpha_{A_1} \wedge \alpha_{B_1} \wedge \mu_{C_1}(z) \tag{5-33}$$

$$R_2: \mu_{C_2'}(Z) = \alpha_{A_2} \wedge \alpha_{B_2} \wedge \mu_{C_2}(z) \tag{5-34}$$

则

$$C' = C_1' \cup C_2' \tag{5-35}$$

即分别从不同的规则得到两个结论,再对所有的结论进行并运算,便得到总的推理结论,其

推理过程可用图 5-21 来表示。

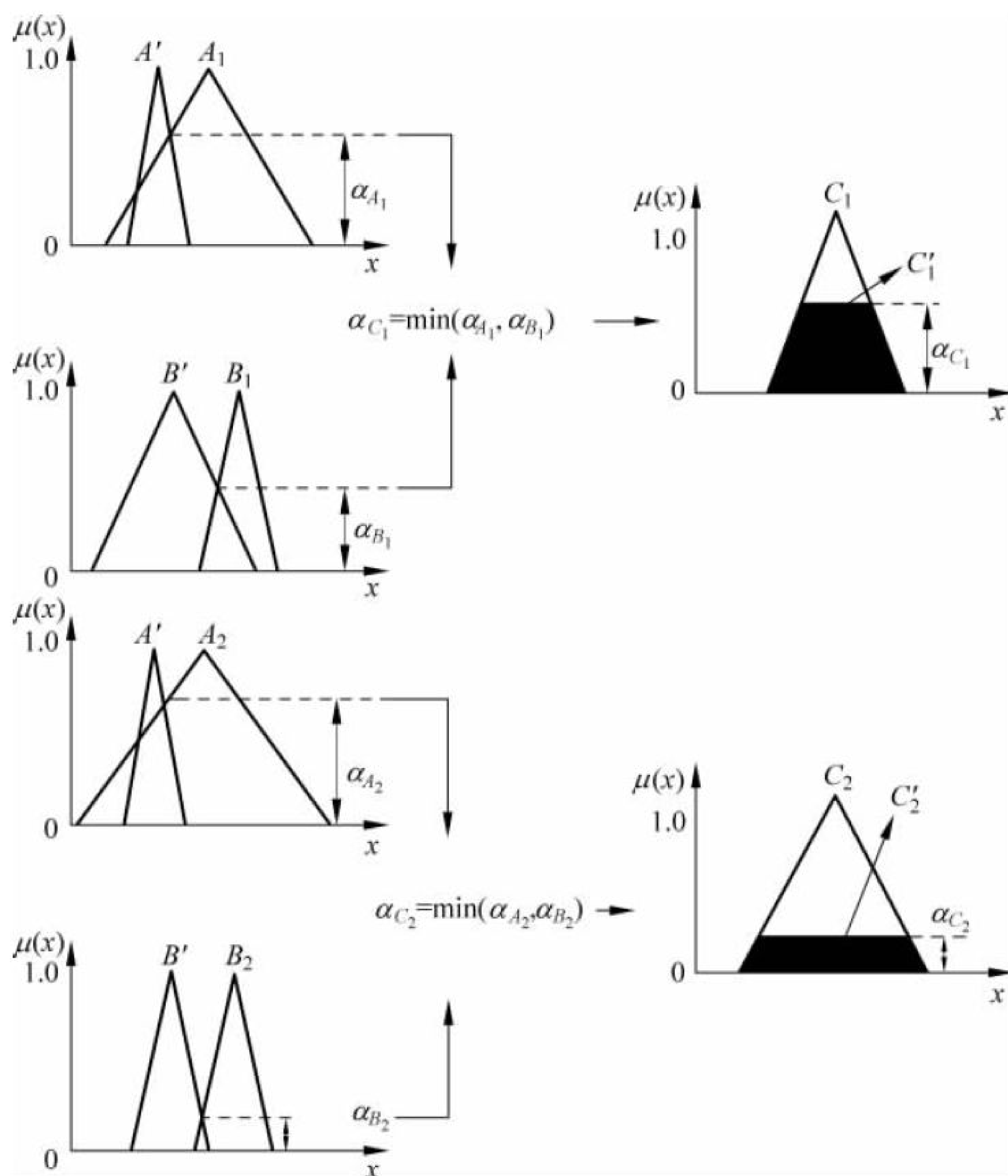


图 5-21　两条二维输入规则的 Mamdani 推理过程

对于多输入多规则的模糊推理，可依据先前提，对大前提中每条模糊条件语句分别进行推理，并将其结果综合成最终推理结果的模糊推理方法，即

$$\mu_{C_i'}(z)=\bigvee_{x\in X}\{\mu_{A'}(x)\wedge[\mu_{A_i}(x)\wedge\mu_{C_i}(z)]\}\cap\bigvee_{y\in Y}\{\mu_{B'}(y)\wedge[\mu_{B_i}(y)\wedge\mu_{C_i}(z)]\}\tag{5-36}$$

可改写为

$$\mu_{C_i'}(z)=\alpha_{A_i}\alpha_{B_i}\alpha_{C_i},\quad i=1,2,\cdots,n\tag{5-37}$$

5.6　数据聚类——模糊聚类

5.6.1　模糊聚类的应用背景

模式识别是一门研究对象描述和分类方法的学科。但在实际应用中，由于数据分布性质不好，致使模式分类时无法精确地定义“规律”或“结构”，而“模糊”的性质对解决该问题提

供了思路。

5.6.2 基于 MATLAB 的 GUI 工具的模糊算法构建——数据模糊化

首先对表 1-1 给定的数据进行分析。很明显，这是一个 3 输入 1 输出系统，3 个输入变量分别为 A、B、C。分别对 3 个输入变量确定各自的输入—输出关系，如表 5-4 所示。

表 5-4 输入与输出关系表

A	B	C	输 出
864.45～1449.58	1641.58～2031.66	2665.9～3405.12	1
2063.54～2949.16	2557.04～3340.14	535.62～1984.98	2
1571.17～1845.59	1575.78～1918.81	1514.98～2396	3
104.8～499.85	3059.54～377.95	2002.33～2462.9	4

1. 输入模糊化

为了更为明确地观察输入和输出的关系，将给定的数据关系利用 MATLAB 工具转换为图形关系，从而得出关系图，然后对 A、B、C 三个变量用模糊化的语言进行描述。

利用 MATLAB 画出变量 A 与输出的关系图，程序代码如下：

```
data = [1739.94  1675.15  2395.96 3
        373.3    3087.05  2429.47 4
        1756.77  1652     1514.98 3
        864.45   1647.31  2665.9 1
        222.85   3059.54  2002.33 4
        877.88   2031.66  3071.18 1
        1803.58  1583.12  2163.05 3
        2352.12  2557.04  1411.53 2
        401.3    3259.94  2150.98 4
        363.34   3477.95  2462.86 4
        1571.17  1731.04  1735.33 3
        104.8    3389.83  2421.83 4
        499.85   3305.75  2196.22 4
        2297.28  3340.14  535.62 2
        2092.62  3177.21  584.32 2
        1418.79  1775.89  2772.9 1
        1845.59  1918.81  2226.49 3
        2205.36  3243.74  1202.69 2
        2949.16  3244.44  662.42 2
        1692.62  1867.5   2108.97 3
        1680.67  1575.78  1725.1 3
        2802.88  3017.11  1984.98 2
        172.78   3084.49  2328.65 4
        2063.54  3199.76  1257.21 2
        1449.58  641.58   3405.12 1
        1651.52  1713.28  1570.38 3
```

```
        341.59    3076.62   2438.63 4
        291.02    3095.68   2088.95 4
        237.63    3077.78   2251.96 4
        ];
stem(data(:,1),data(:,4),' * ','r'); (绘制输入输出的二维杆图)
title('数据 A 与输出类型的关系图');
xlabel('数据 A');
ylabel('输出分类');
```

运行程序,得到如图 5-22 所示的数据 A 与输出类型的关系图。

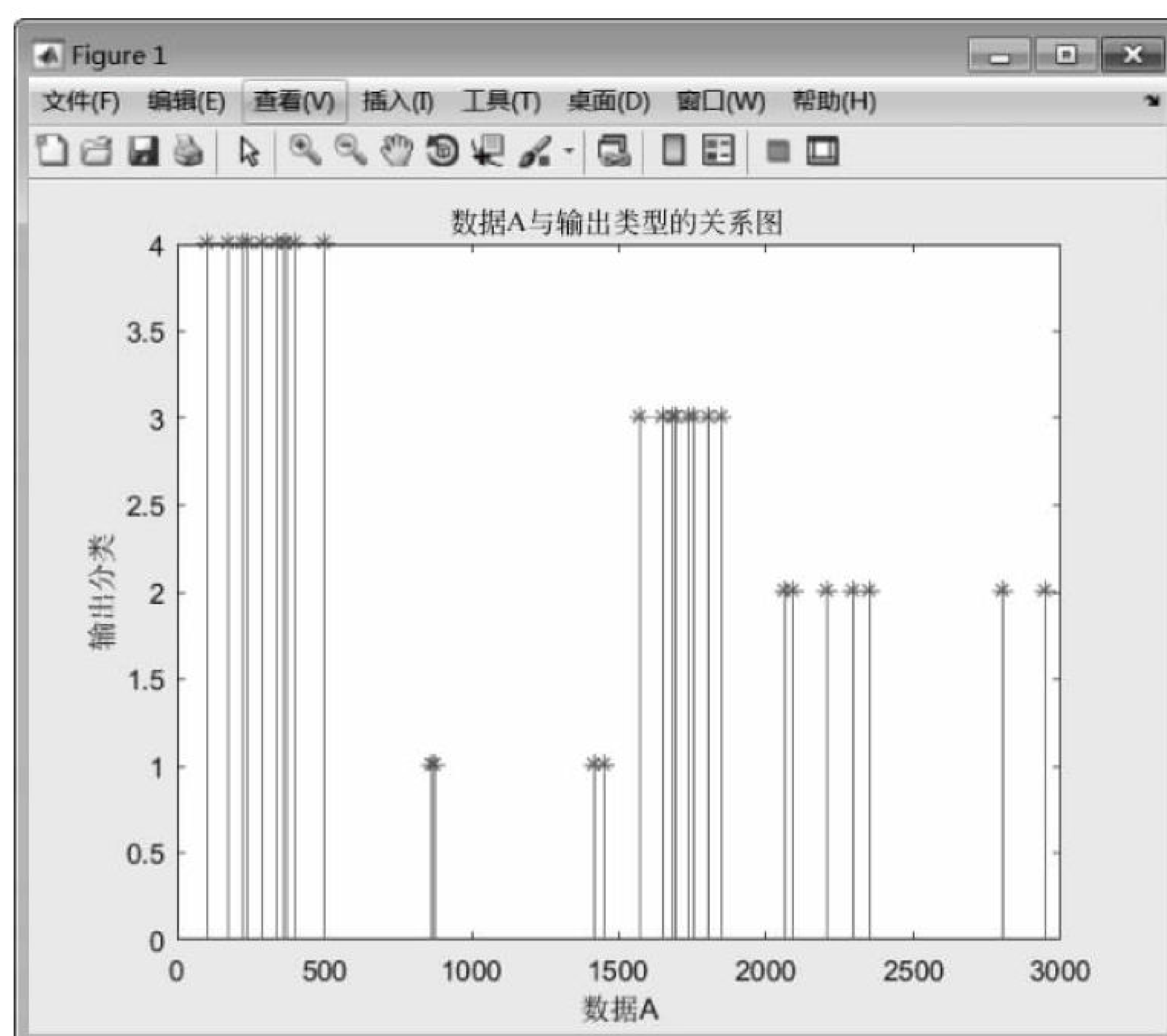


图 5-22 数据 A 与输出类型的关系图

从输入和输出的关系图中可以看出,可将 A 模糊化为四种状态,分别为小、偏小、偏大、大,从而完成对输入 A 的模糊化。

同理,利用 MATLAB 画出变量 B、变量 C 与输出的关系图,程序代码如下:

```
data = [1739.94   1675.15   2395.96 3
        373.3     3087.05   2429.47 4
        1756.77   1652      1514.98 3
        864.45    1647.31   2665.9 1
        222.85    3059.54   2002.33 4
        877.88    2031.66   3071.18 1
        1803.58   1583.12   2163.05 3
        2352.12   2557.04   1411.53 2
        401.3     3259.94   2150.98 4
```

```
        363.34    3477.95   2462.86 4
        1571.17   1731.04   1735.33 3
        104.8     3389.83   2421.83 4
        499.85    3305.75   2196.22 4
        2297.28   3340.14   535.62 2
        2092.62   3177.21   584.32 2
        1418.79   1775.89   2772.9 1
        1845.59   1918.81   2226.49 3
        2205.36   3243.74   1202.69 2
        2949.16   3244.44   662.42 2
        1692.62   1867.5    2108.97 3
        1680.67   1575.78   1725.1 3
        2802.88   3017.11   1984.98 2
        172.78    3084.49   2328.65 4
        2063.54   3199.76   1257.21 2
        1449.58   1641.58   3405.12 1
        1651.52   1713.28   1570.38 3
        341.59    3076.62   2438.63 4
        291.02    3095.68   2088.95 4
        237.63    3077.78   2251.96 4
        ];
subplot(1,2,1);    (生成 1 * 2 个子图,当前激活第一个子图)
stem(data(:,2),data(:,4),'g','s');
title('数据 B 与输出类型的关系图');
xlabel('数据 B');
ylabel('输出分类');
subplot(1,2,2);
stem(data(:,3),data(:,4),'b','o');
title('数据 C 与输出类型的关系图');
xlabel('数据 C');
ylabel('输出分类');
```

运行程序后,得到的关系图如图 5-23 所示。

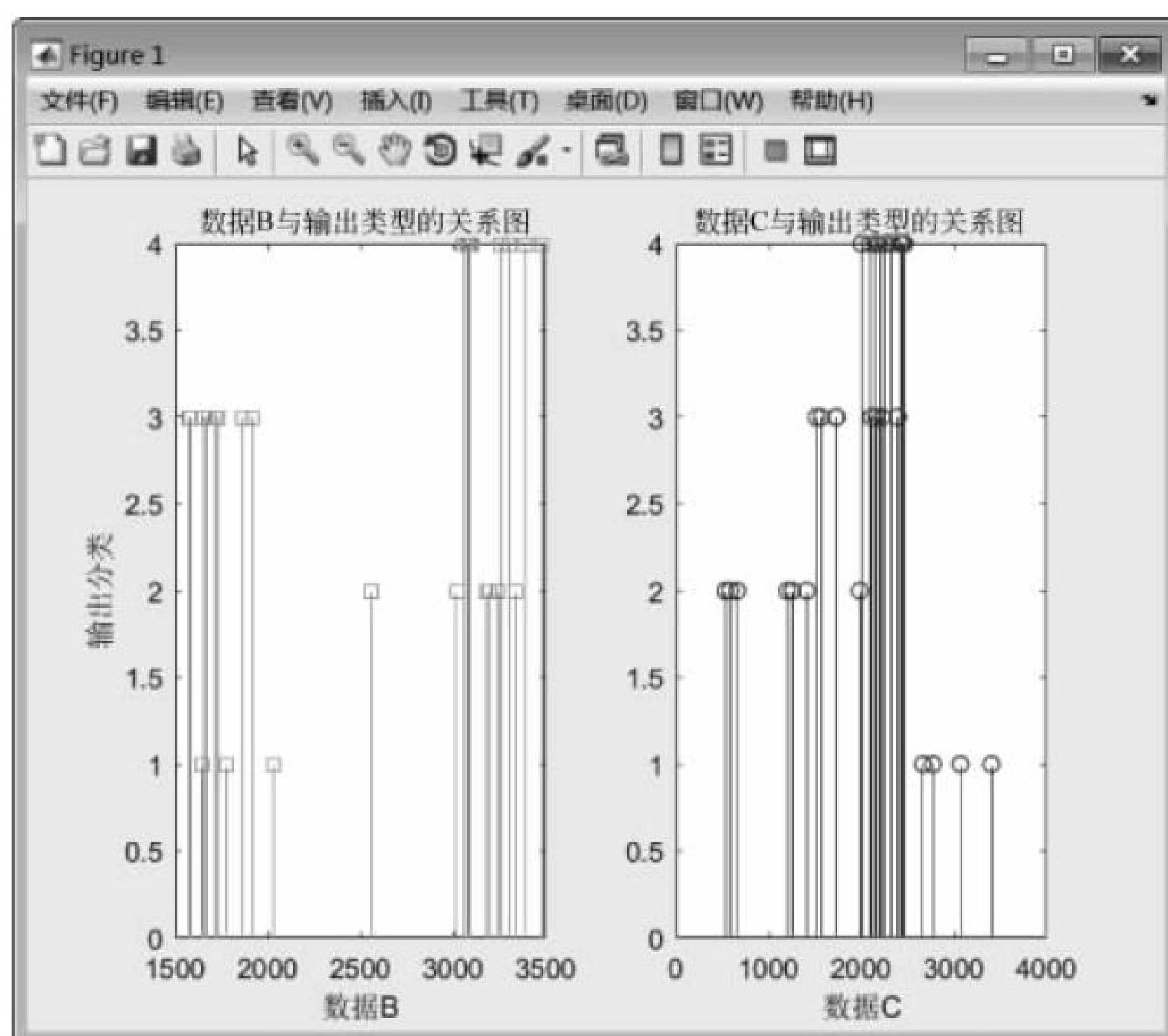


图 5-23　变量 B、C 与输出类型的关系图

从图 5-23 中可以看出，输入 B 和输出的关系不是非常明显，在这里试将 B 模糊化为小、中、大三种状态，从而完成对输入 B 的模糊化；而输入 C 和输出的关系相对明朗，在此试将 C 模糊化为小、偏小、偏大、大四个模糊语言，从而完成了对输入 C 的模糊化。

2. 隶属度函数的选择

隶属度函数可以是任意形状的曲线，在此选择梯形隶属度函数，其格式如下：

```
y = trapmf(x,[a b c d])
```

其中，参数 a 和 d 确定梯形的“脚”；而参数 b 和 c 确定梯形的“肩膀”。

各输入信号隶属度函数参数的选择如下：

```
A:  samll     [0 0 499 864]
    psmall    [499 864 1450 1571]
    pbig      [1450 1571 1846 2064]
    big       [1846 2064 3000 3000]
B:  small     [1400 1400 2032 2557]
    mid       [2032 2557 3017 3060]
    big       [3017 3060 3500 3500]
C:  small     [0 350 1412 1515]
    psmall    [1412 1515 1735 2002]
    pbig      [1735 2002 2463 2666]
    big:      [2463 2666 3500 3500]
```

3. 模糊规则的建立

建立模糊规则如下：

```
A(偏小)and B(小)and C (大),输出为 1
A(大) and B(大)and C (小),输出为 2
A(大) and B(中)and C (小),输出为 2
A(偏大)and B(小)and C(偏小),输出为 3
A(偏大)and B(小)and C(偏大),输出为 3
A(小) and B(大)and C(偏大),输出为 4
```

5.6.3　基于 MATLAB 的 GUI 工具的模糊算法构建——FIS 实现

首先在 FIS 界面设置模糊运算方式，选择结果如图 5-24 所示。

其中，解模糊选择 MOM 方式。然后编辑输入变量和输出变量的隶属度函数，如图 5-25 所示。

输入模糊规则表，如图 5-26 所示。

此时，用户可以从输出曲面观测器中观测整个论域上输出变量与输入变量的关系。图 5-27 为输入变量 A、B 与输出变量间的关系图。

至此模糊分类系统设计完成。

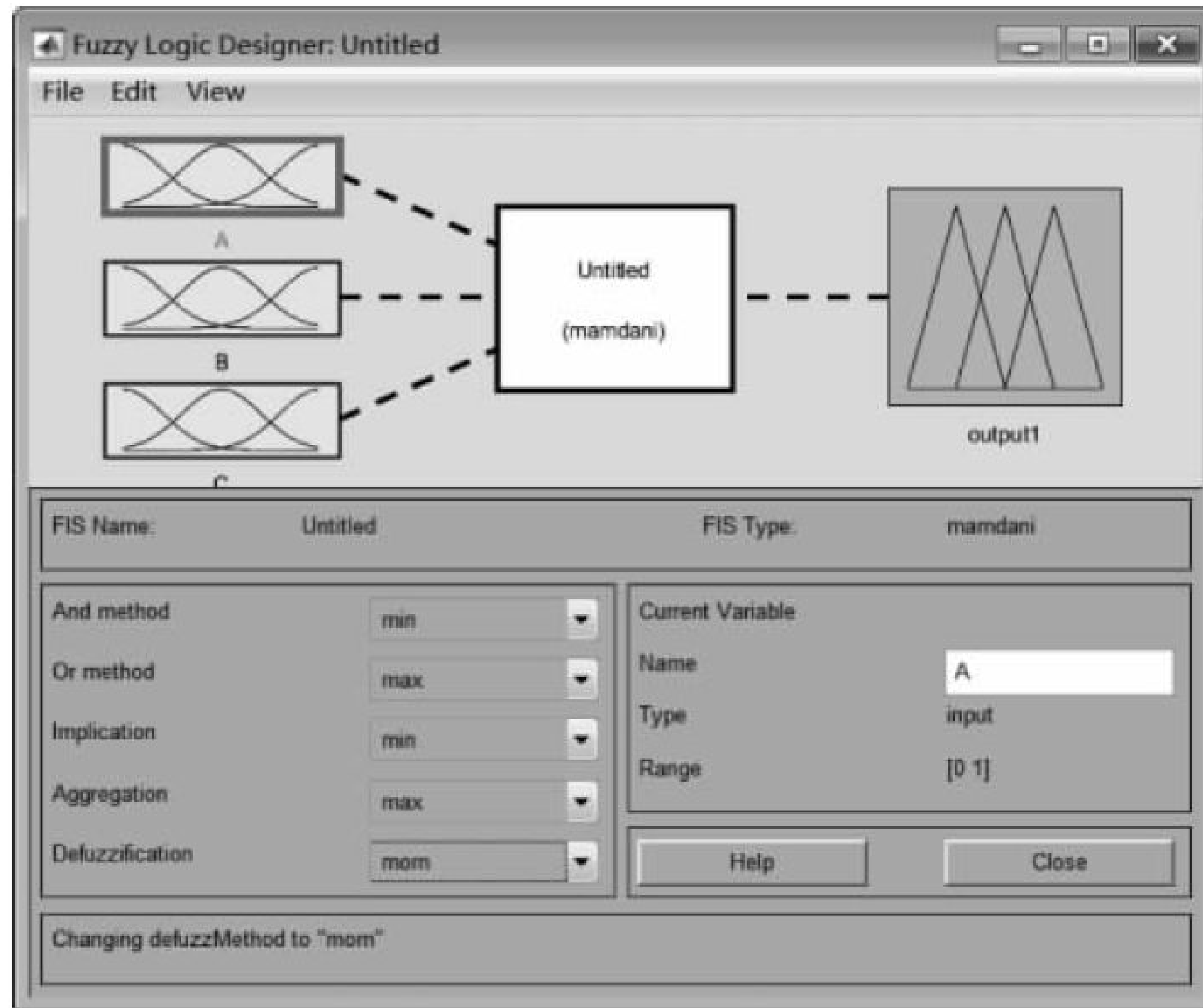
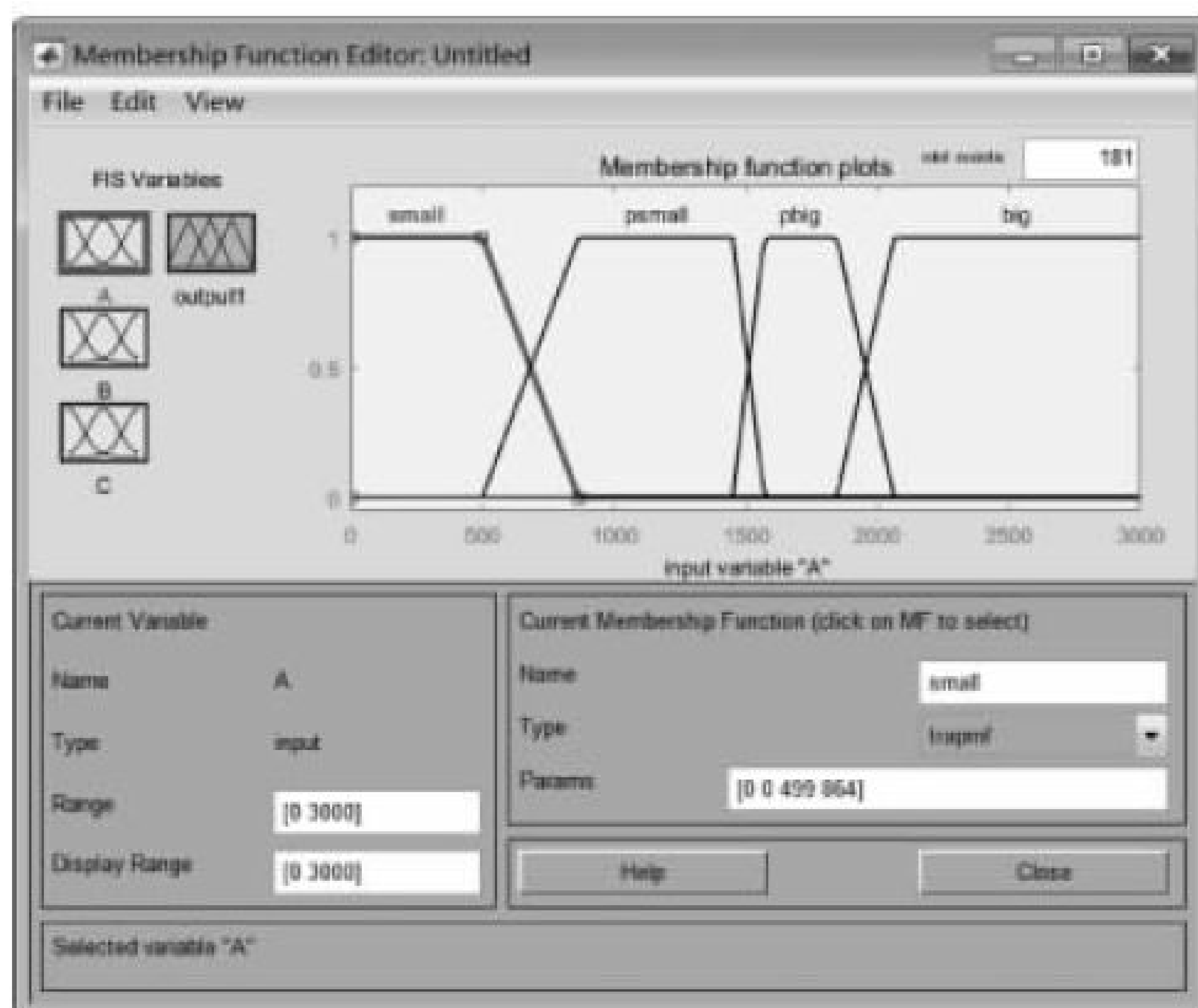
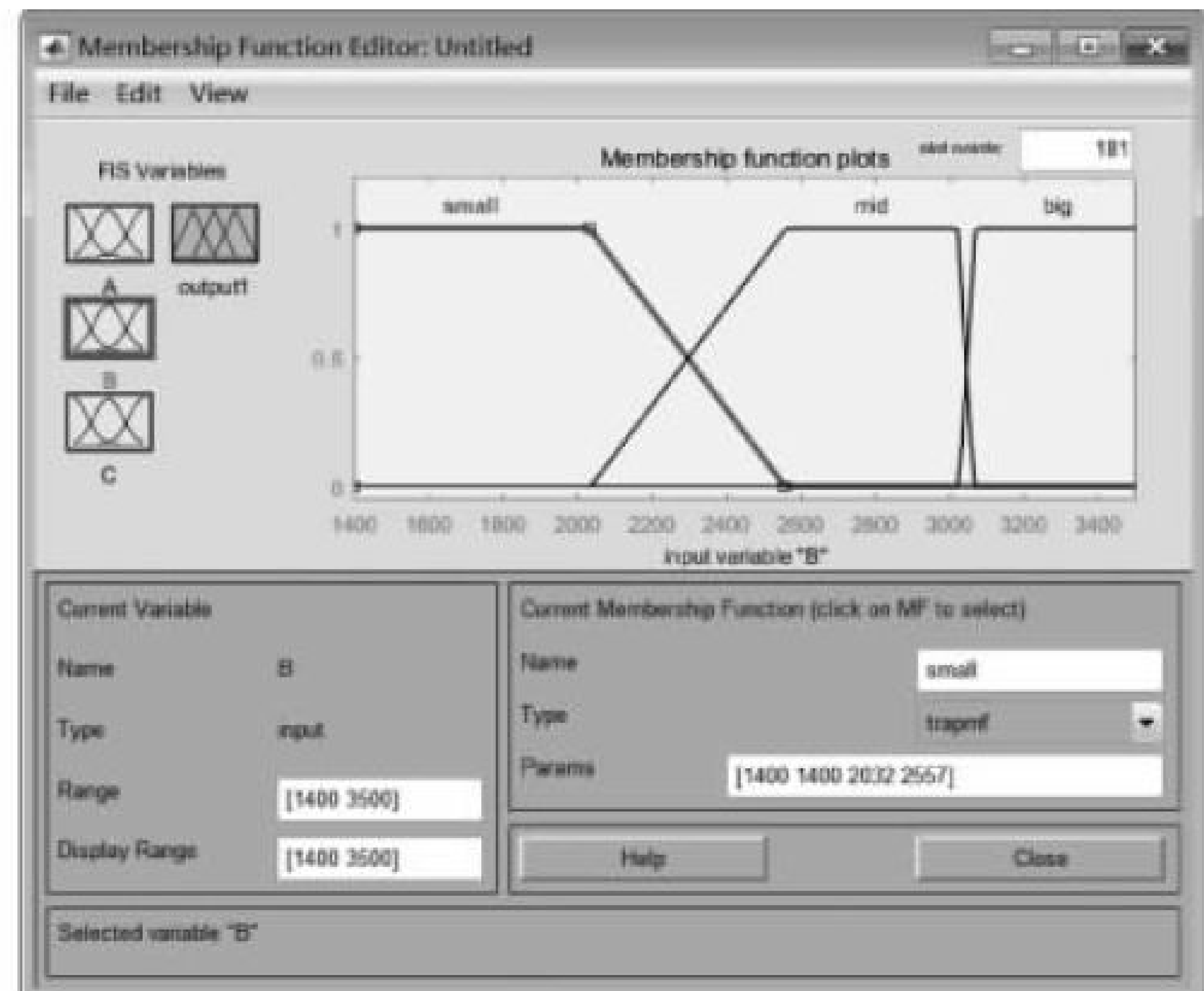


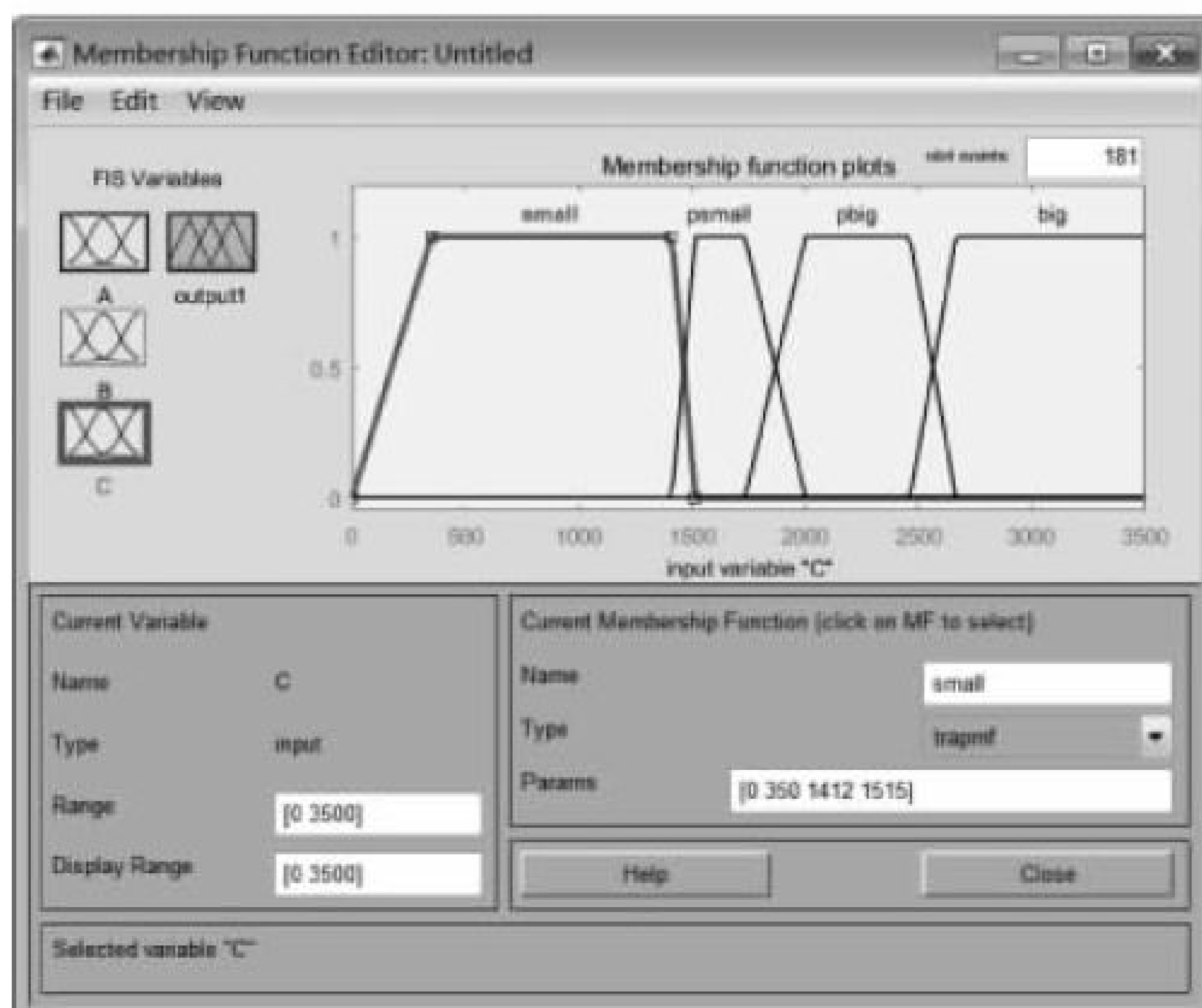
图 5-24 模糊运算方式的选择



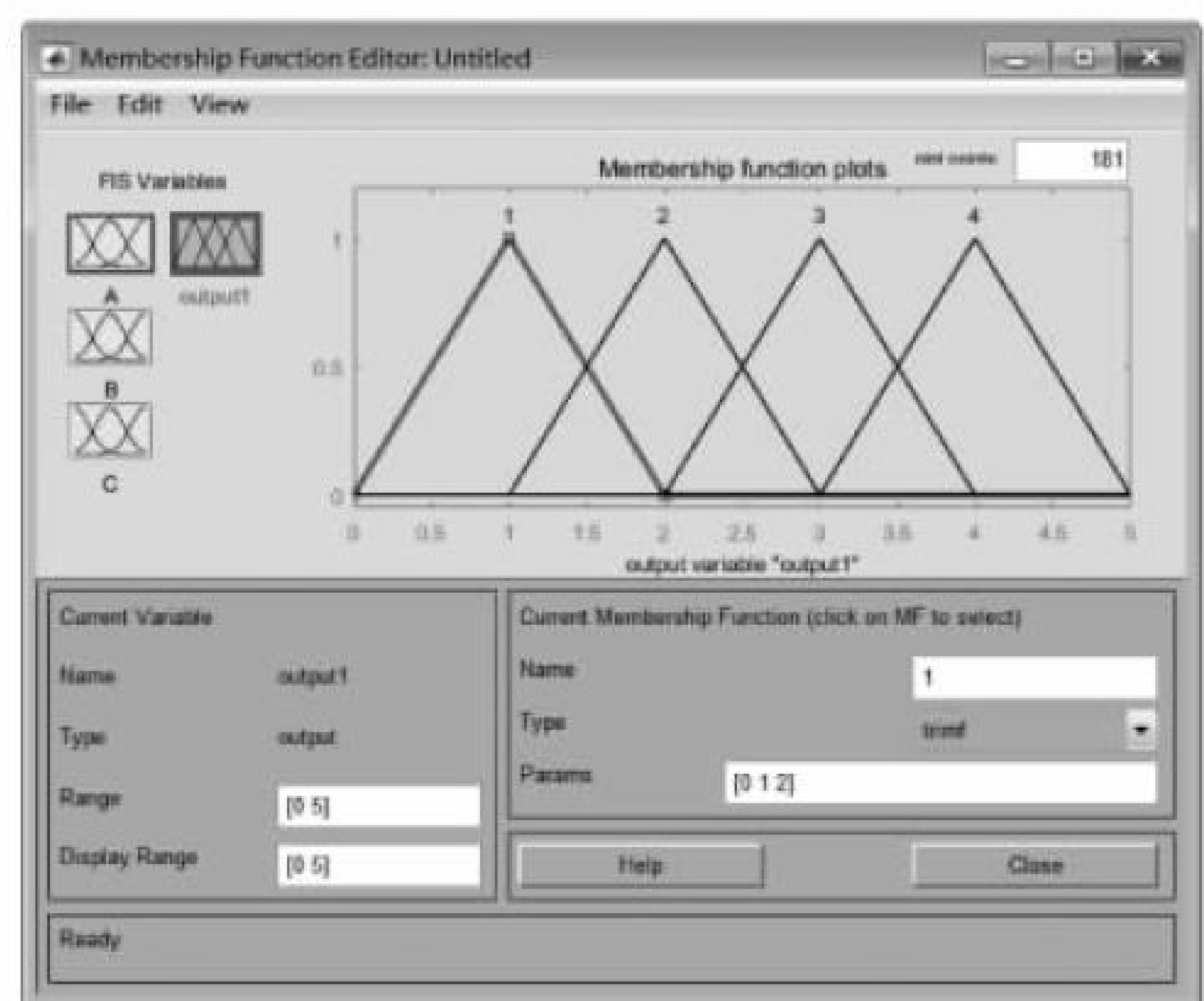
(a) 输入变量A的隶属度函数



(b) 输入变量B的隶属度函数



(c) 输入变量C的隶属度函数



(d) 输出变量Type的隶属度函数

图 5-25 输入变量和输出变量的隶属度函数

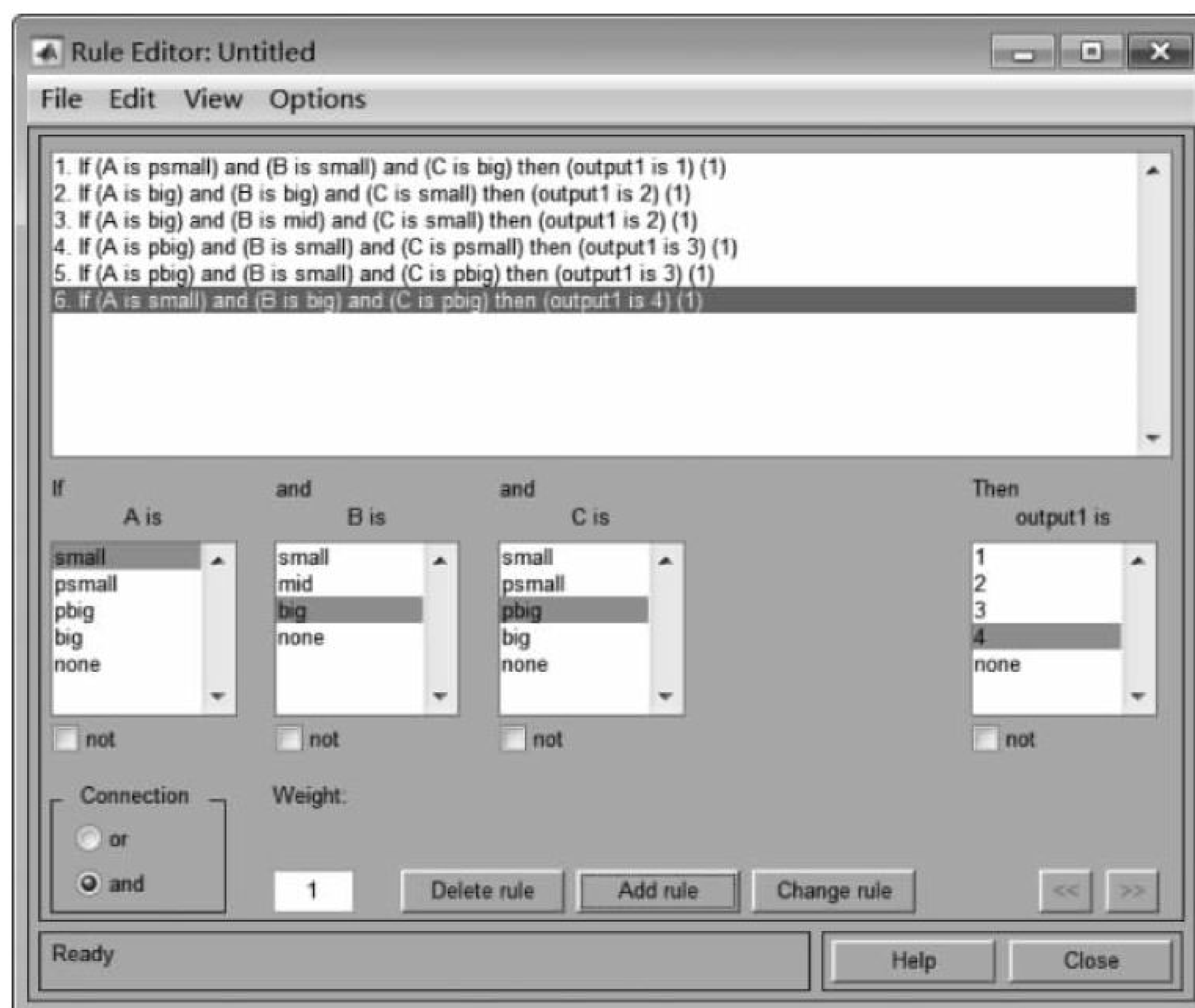


图 5-26 输入模糊规则表

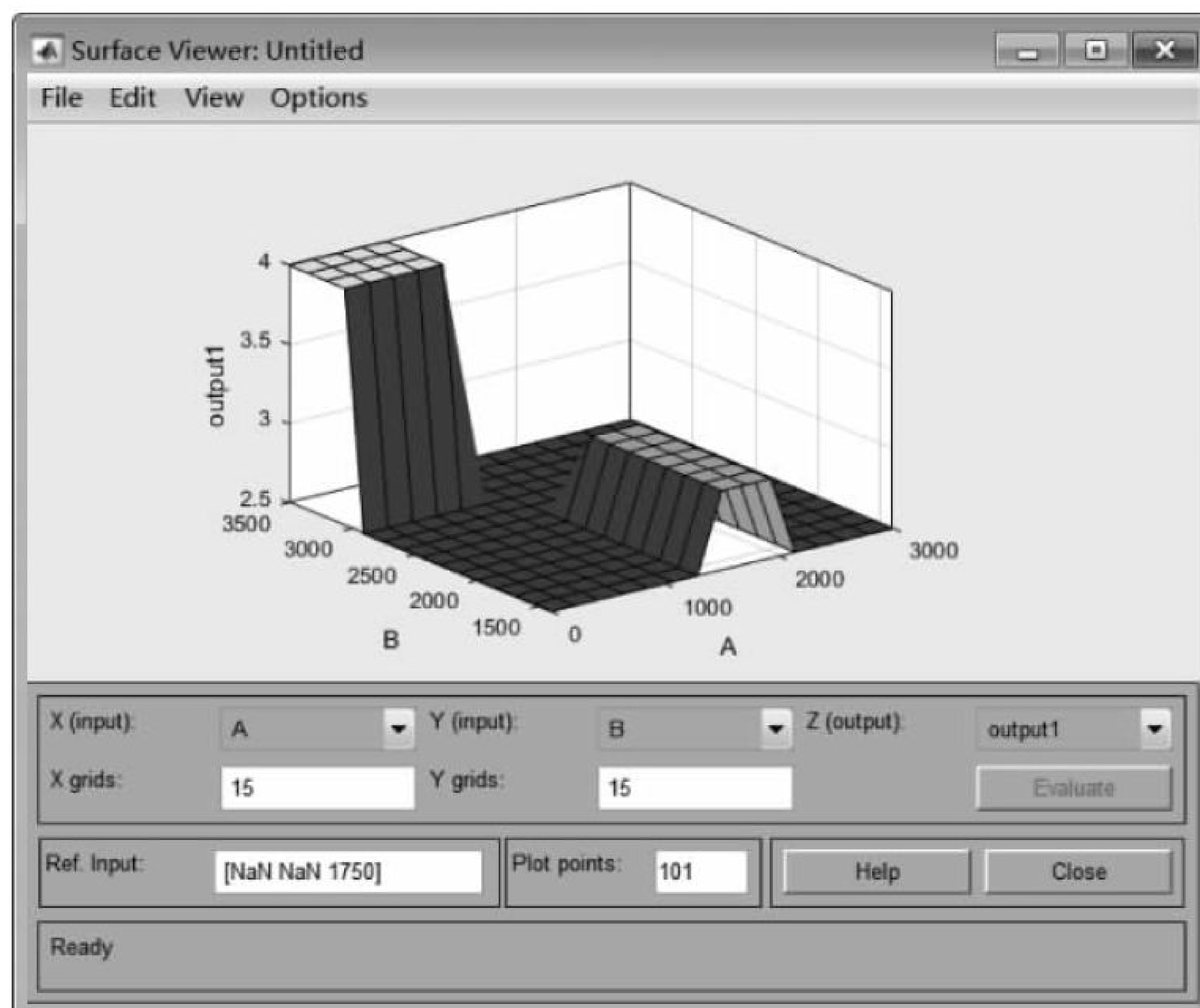


图 5-27 输入变量 A、B 与输出变量间的关系图

5.6.4 系统结果分析

打开规则观测器,界面如图 5-28 所示。

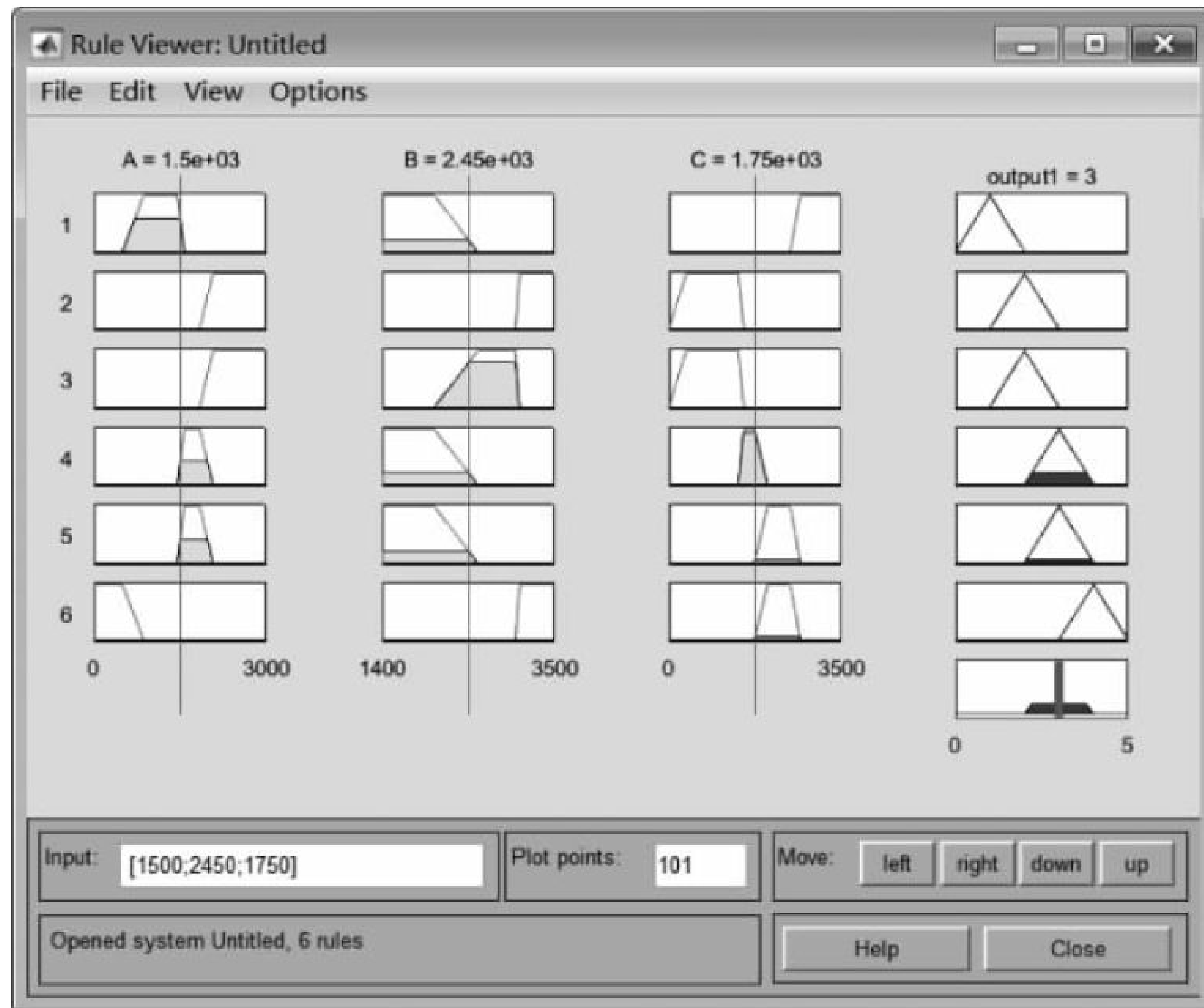


图 5-28 规则观测器

在 Input 文本框输入样本,即可从输出端口得到相应的分类结果,如图 5-29 所示。

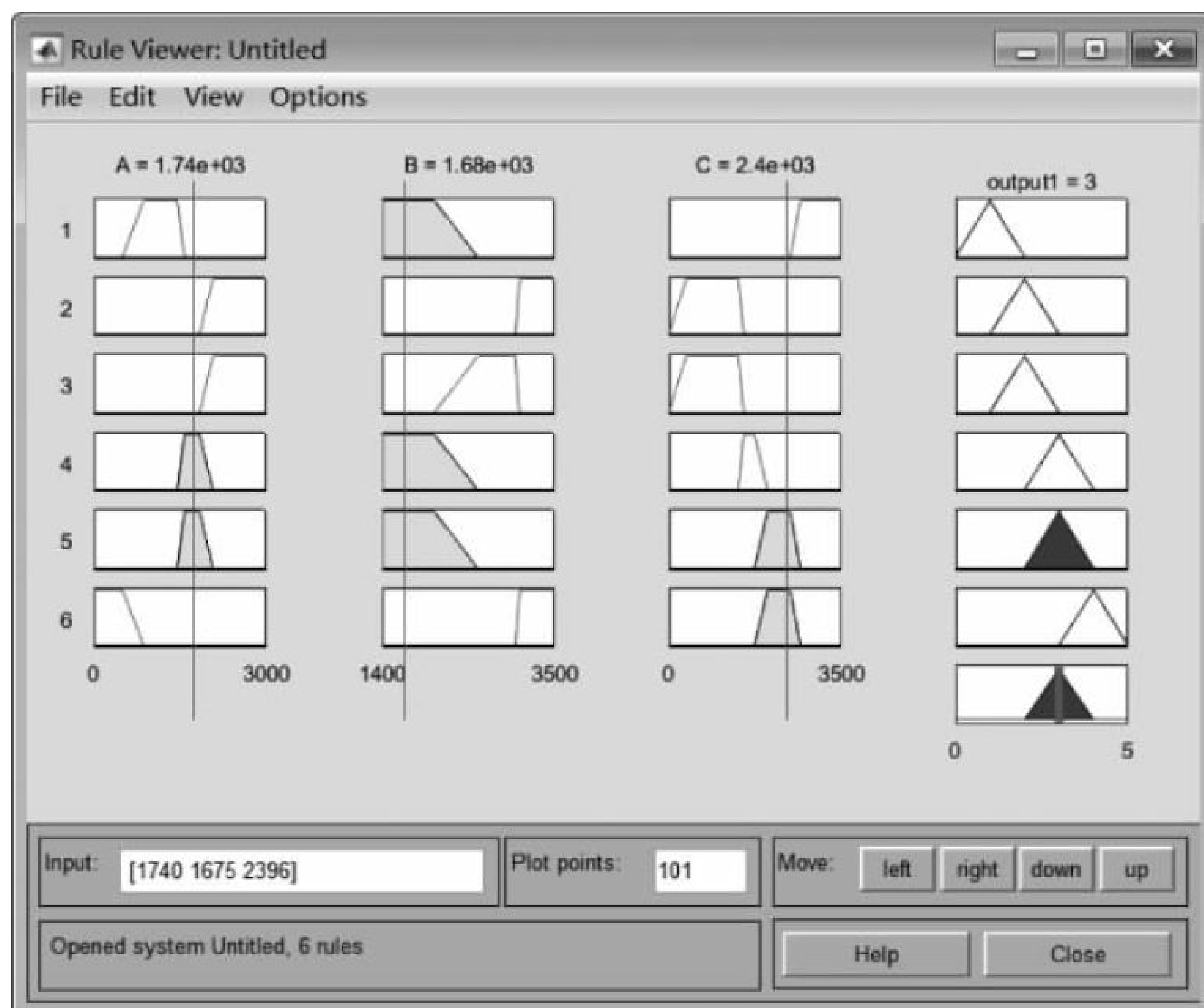


图 5-29 测试数据窗口

输入样本数据[1739.94,1675.15,2395.96]后，即可得分类值3。按照上述方法测试数据，结果如表5-5所示。

表5-5 模糊系统分类结果

序 号	A	B	C	目标分类结果	模糊分类系统测试结果
1	1739.94	1675.15	2395.96	3	3
2	373.3	3087.05	2429.47	4	4
3	1756.77	1652	1514.98	3	3
4	864.45	1647.31	2665.9	1	1
5	222.85	3059.54	2002.33	4	4
6	877.88	2031.66	3071.18	1	1
7	1803.58	1583.12	2163.05	3	3
8	2352.12	2557.04	1411.53	2	2
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
11	1571.17	1731.04	1735.33	3	3
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
16	1418.79	1775.89	2772.9	1	1
17	1845.59	1918.81	2226.49	3	3
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
20	1692.62	1867.5	2108.97	3	3
21	1680.67	1575.78	1725.1	3	3
22	2802.88	3017.11	1984.98	2	2.5
23	172.78	3084.49	2328.65	4	4
24	2063.54	3199.76	1257.21	2	2
25	1449.58	1641.58	3405.12	1	1
26	1651.52	1713.28	1570.38	3	3
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4
30	1702.8	1639.79	2068.74		3
31	1877.93	1860.96	1975.3		3
32	867.81	2334.68	2535.1		1
33	1831.49	1713.11	1604.68		3
34	460.69	3274.77	2172.99		4
35	2374.98	3346.98	975.31		2
36	2271.89	3482.97	946.7		2
37	1783.64	1597.99	2261.31		3
38	198.83	3250.45	2445.08		4
39	1494.63	2072.59	2550.51		1

续表

序 号	A	B	C	目标分类结果	模糊分类系统测试结果
40	1597.03	1921.52	2126.76		3
41	1598.93	1921.08	1623.33		3
42	1243.13	1814.07	3441.07		1
43	2336.31	2640.26	1599.63		2.5
44	354	3300.12	2373.61		4
45	2144.47	2501.62	591.51		2
46	426.31	3105.29	2057.8		4
47	1507.13	1556.89	1954.51		3
48	343.07	3271.72	2036.94		4
49	2201.94	3196.22	935.53		2
50	2232.43	3077.87	1298.87		2
51	1580.1	1752.07	2463.04		3
52	1962.4	1594.97	1835.95		3
53	1495.18	1957.44	3498.02		1
54	1125.17	1594.39	2937.73		1
55	24.22	3447.31	2145.01		4
56	1269.07	1910.72	2701.97		1
57	1802.07	1725.81	1966.35		3
58	1817.36	1927.4	2328.79		3
59	1860.45	1782.88	1875.13		3

从系统的分类结果可知,错误率为1/29≈3.4%。用户可修改输入数据的隶属度函数、模糊控制规则表,进一步降低分类错误率。

5.6.5 结论

本节利用MATLAB中的GUI工具箱来建立模糊控制系统,基本实现了酒瓶颜色的分类,模糊聚类是一种软分类方法,用于解决没有明显界线的类别。其中模糊规则是影响最后聚类结果最关键的因素。当分类效果不好时,可以通过修改隶属度函数和模糊规则来实现最优分类。

5.7 数据聚类——模糊 *C* 均值聚类

5.7.1 模糊 *C* 均值聚类的应用背景

传统的聚类分析是一种硬划分(crisp partition),它把每个待辨识的对象严格地划分到某类中,具有“非此即彼”的性质,因此这种类别划分的界限是分明的。然而实际上大多数对象并没有严格的属性,它们在性质和类属方面存在着中介性,具有“亦此亦彼”的性质,因此

适合进行软划分。Zadeh 提出的模糊集理论为这种软划分提供了有力的分析工具,人们开始用模糊方法来处理聚类问题,并称为模糊聚类分析。模糊聚类得到了样本属于各个类别的不确定性程度,表达了样本类属的中介性,建立起了样本对于类别的不确定性的描述,能更客观地反映现实世界,从而成为聚类分析研究的主流。

在基于目标函数的聚类算法中模糊 C 均值(fuzzy c-means,FCM)类型算法的理论最为完善,应用最为广泛。

5.7.2 模糊 C 均值算法

1. 模糊 C 均值聚类的准则

设 $x_i(i=1,2,\cdots,n)$是 n 个样本组成的样本集合,c 为预定的类别数目,$\mu_j(x_i)$是第 i 个样本对于第 j 类的隶属度函数。用隶属度函数定义的聚类损失函数可以写为

$$J_f=\sum_{j=1}^{c}\sum_{i=1}^{n}[\mu_j(x_i)]^b\ \|x_i-m_j\|^2 \tag{5-38}$$

其中,$b>1$,是一个可以控制聚类结果的模糊程度的常数。

在不同的隶属度定义方法下最小化聚类损失函数,就得到不同的模糊聚类方法。其中最有代表性的是模糊 C 均值方法,它要求一个样本对于各个聚类的隶属度之和为 1,即

$$\sum_{j=1}^{c}\mu_j(x_i)=1,\quad i=1,2,\cdots,n \tag{5-39}$$

2. 模糊 C 均值算法步骤

(1) 设定聚类数目 c 和加权指数 b。

J. C. Bezdek 根据经验,认为 b 取 2 最合适。

Cheung 和 Chen 从汉字识别的应用背景得出 b 的最佳取值应在 1.25～1.75。

Bezdek 和 Hathaway 等人从算法收敛性角度着手,得出 b 的取值与样本数目 n 有关的结论,建议 b 的取值要大于 $n/(n-2)$。

Pal 等人从聚类有效性方面的实验研究得到 b 的最佳选取区间为[1.5,2.5],在不做特殊要求下可取区间中值 $b=2$。

(2) 初始化各个聚类中心 m_i

$$m_i=\frac{1}{N_i}\sum_{y\in\Gamma_i}y \tag{5-40}$$

式中,N_i是第 i 聚类 Γ_i中的样本数目。

(3) 重复下面的运算,直到各个样本的隶属度值稳定。

用当前的聚类中心根据下式计算隶属度函数:

$$\mu_j(x_i)=\frac{\left(\dfrac{1}{x_i}-m_j^2\right)^{\frac{1}{b-1}}}{\sum\limits_{k=1}^{c}\left(\dfrac{1}{x_i}-m_k^2\right)^{\frac{1}{b-1}}} \tag{5-41}$$

用当前的隶属度函数按下式更新计算各类聚类中心:

$$m_j = \frac{\sum_{i=1}^{n} [\mu_j(x_i)]^b x_i}{\sum_{i=1}^{n} [\mu_j(x_i)]^b} \tag{5-42}$$

当模糊 C 均值算法收敛时，就得到了各类的聚类中心和各个样本对于各类的隶属度值，从而完成了模糊聚类划分。如果需要，还可以将模糊聚类结果进行解模糊，即用一定的规则把模糊聚类划分转化为确定性分类。

5.7.3 模糊 C 均值聚类的 MATLAB 实现

这里还是采用表 1-2 所示的数据。

1. MATLAB 模糊 C 均值数据聚类识别函数

在 MATLAB 中($b=2$)，只要直接调用如下程序即可实现模糊 C 均值聚类：

```
[Center,U,obj_fcn] = fcm(data,cluster_n)
```

其中，data：要聚类的数据集合，每一行为一个样本；cluster_n：聚类数；Center：最终的聚类中心矩阵，每一行为聚类中心的坐标值；U：最终的模糊分区矩阵；obj_fcn：在迭代过程中的目标函数值。

注意：在使用上述方法时，要根据中心坐标 Center 的特点分清楚每一类中心所代表的实际中的哪一类，然后才能准确地将待聚类的各方案准确地分为各自所属的类别；否则，就会出现张冠李戴的现象。

2. MATLAB 图形显示聚类模式

使用命令[Center,U,obj_fcn] = fcm(data,4)进行聚类后，可调用 MATLAB 图形窗口显示聚类结果，命令格式如下：

```
maxU = max(U);                          % 最大隶属度
index1 = find(U(1,:) == maxU)           % 找到属于第一类的点
index2 = find(U(2,:) == maxU)           % 找到属于第二类的点
index3 = find(U(3,:) == maxU)           % 找到属于第三类的点
index4 = find(U(4,:) == maxU)           % 找到属于第四类的点
```

为了提高图形的区分度，添加如下命令：

```
line(data(index1,1),data(index1,2),data(index1,3),'linestyle','none','marker','*','color','g');
line(data(index2,1),data(index2,2),data(index2,3),'linestyle','none','marker','*','color','r');
line(data(index3,1),data(index3,2),data(index3,3),'linestyle','none','marker','+','color','b');
line(data(index4,1),data(index4,2),data(index4,3),'linestyle','none','marker','+','color','y');
```

3. MATLAB 实现模糊 C 均值聚类

实现模糊 C 均值聚类的代码如下：

```
clear all;
data = [1739.94  1675.15  2395.96
        373.3    3087.05  2429.47
        1756.77  1652     1514.98
        864.45   1647.31  2665.9
        222.85   3059.54  2002.33
        877.88   2031.66  3071.18
        1803.58  1583.12  2163.05
        2352.12  2557.04  1411.53
        401.3    3259.94  2150.98
        363.34   3477.95  2462.86
        1571.17  1731.04  1735.33
        104.8    3389.83  2421.83
        499.85   3305.75  2196.22
        2297.28  3340.14  535.62
        2092.62  3177.21  584.32
        1418.79  1775.89  2772.9
        1845.59  1918.81  2226.49
        2205.36  3243.74  1202.69
        2949.16  3244.44  662.42
        1692.62  1867.5   2108.97
        1680.67  1575.78  1725.1
        2802.88  3017.11  1984.98
        172.78   3084.49  2328.65
        2063.54  3199.76  1257.21
        1449.58  1641.58  3405.12
        1651.52  1713.28  1570.38
        341.59   3076.62  2438.63
        291.02   3095.68  2088.95
        237.63   3077.78  2251.96
        1702.8   1639.79  2068.74
        1877.93  1860.96  1975.3
        867.81   2334.68  2535.1
        1831.49  1713.11  1604.68
        460.69   3274.77  2172.99
        2374.98  3346.98  975.31
        2271.89  3482.97  946.7
        1783.64  1597.99  2261.31
        198.83   3250.45  2445.08
        1494.63  2072.59  2550.51
        1597.03  1921.52  2126.76
        1598.93  1921.08  1623.33
        1243.13  1814.07  3441.07
        2336.31  2640.26  1599.63
        354      3300.12  2373.61
        2144.47  2501.62  591.51
        426.31   3105.29  2057.8
        1507.13  1556.89  1954.51
        343.07   3271.72  2036.94
        2201.94  3196.22  935.53
```

```
        2232.43  3077.87  1298.87
        1580.1   1752.07  2463.04
        1962.4   1594.97  1835.95
        1495.18  1957.44  3498.02
        1125.17  1594.39  2937.73
        24.22    3447.31  2145.01
        1269.07  1910.72  2701.97
        1802.07  1725.81  1966.35
        1817.36  1927.4   2328.79
        1860.45  1782.88  1875.13
        ];
[center,U,obj_fcn] = fcm(data,4);
plot3(data(:,1),data(:,2),data(:,3),'o');
grid;
maxU = max(U);
index1 = find(U(1,:) == maxU)
index2 = find(U(2,:) == maxU)
index3 = find(U(3,:) == maxU)
index4 = find(U(4,:) == maxU)
line(data(index1,1),data(index1,2),data(index1,3),'linestyle','none','marker','*','color','g');
line(data(index2,1),data(index2,2),data(index2,3),'linestyle','none','marker','*','color','r');
line(data(index3,1),data(index3,2),data(index3,3),'linestyle','none','marker','+','color','b');
line(data(index4,1),data(index4,2),data(index4,3),'linestyle','none','marker','+','color','y');
title('模糊C均值聚类分析图');
xlabel('第一特征坐标');
ylabel('第二特征坐标');
zlabel('第三特征坐标');
```

5.7.4 模糊 C 均值聚类结果分析

运行 MATLAB 程序,数据的模糊 C 均值聚类分析数据如下:

```
Iteration count = 1,obj. fcn = 28484303.583307
Iteration count = 2,obj. fcn = 22894174.219903
Iteration count = 3,obj. fcn = 22492974.034424
Iteration count = 4,obj. fcn = 20879539.602697
Iteration count = 5,obj. fcn = 14444987.068964
Iteration count = 6,obj. fcn = 8322567.664727
Iteration count = 7,obj. fcn = 7551351.839018
Iteration count = 8,obj. fcn = 7439273.677928
Iteration count = 9,obj. fcn = 7421451.003657
Iteration count = 10,obj. fcn = 7417960.721127
Iteration count = 11,obj. fcn = 7417133.213718
Iteration count = 12,obj. fcn = 7416918.432660
Iteration count = 13,obj. fcn = 7416860.845351
Iteration count = 14,obj. fcn = 7416845.240472
Iteration count = 15,obj. fcn = 7416840.997724
```

```
Iteration count = 16,obj. fcn = 7416839.842995
Iteration count = 17,obj. fcn = 7416839.528623
Iteration count = 18,obj. fcn = 7416839.443030
Iteration count = 19,obj. fcn = 7416839.419726
Iteration count = 20,obj. fcn = 7416839.413381
Iteration count = 21,obj. fcn = 7416839.411653
Iteration count = 22,obj. fcn = 7416839.411183
Iteration count = 23,obj. fcn = 7416839.411055
Iteration count = 24,obj. fcn = 7416839.411020
Iteration count = 25,obj. fcn = 7416839.411010
index1 =
4   6   16   25   32   39   42   53   54   56
index2 =
2   5   9   10   12   13   23   27   28   29   34   38   44   46   48   55
index3 =
8   14   15   18   19   22   24   35   36   43   45   49   50
index4 =
1   3   7   11   17   20   21   26   30   31   33   37   40   41   47   51   52   57   58   59
```

分类结果图如图 5-30 所示。

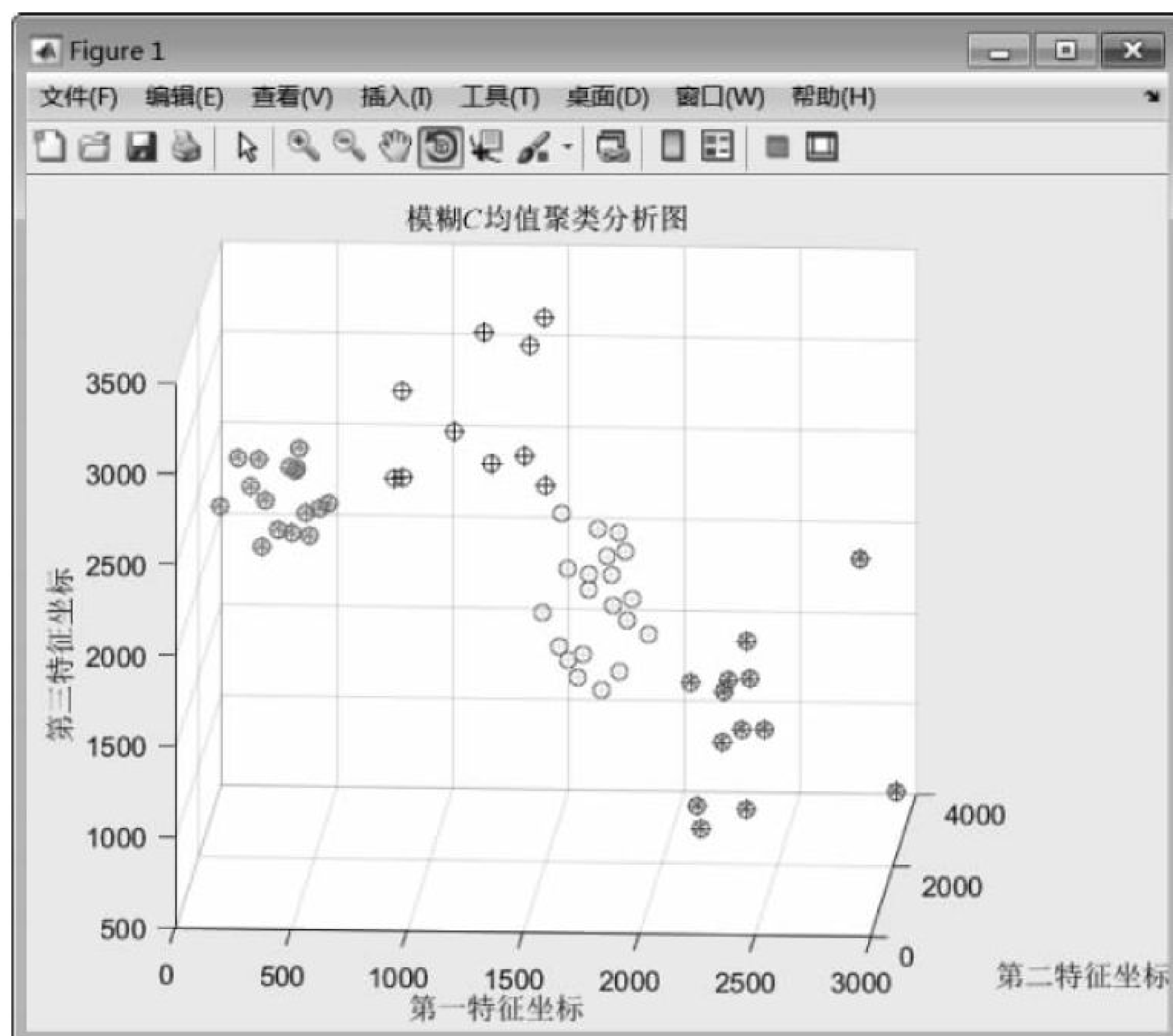


图 5-30　分类结果图

经过对比发现，用模糊 C 均值进行聚类分析的分类结果与给定结果完全吻合。

5.7.5 结论

模糊 C 均值聚类算法是目前比较常用的模糊聚类算法,它有着完善的理论和深厚的数学基础。当数据的结果簇是密集的,且簇与簇之间区分明显时,它的效果较好。而且该算法是相对可伸缩和高效率的,因为算法的复杂度是 $O(ncb)$,其中,n 是用户对象的个数,c 是聚类的数目,b 是迭代的次数。

然而模糊 C 均值聚类算法也有不少缺点:

(1) 模糊 C 均值聚类算法对孤立点数据比较敏感。

(2) 模糊 C 均值聚类算法需要事先指定聚类数目 c 和模糊加权指数 m,而 c 和 m 直接影响着聚类的结果。

(3) 由于模糊聚类的目标函数是非凸的,而模糊 C 均值聚类类型算法又是迭代爬山的,因此容易陷入局部极值点或鞍点,而得不到最优解。

5.8 数据聚类——模糊 ISODATA 聚类

5.8.1 模糊 ISODATA 聚类的应用背景

G. H. Ball 与 D. J. Hall 于 1965 年提出的 ISODATA 算法是一个通过逐步修改聚类中心的个数与位置来达到分类目的的集群算法,后来不断有人提出它的各种改进算法,其中包括 Ball 和 Hall 1967 年提出的改进算法、CLASS、ASP 等。1974 年 J. C. Dunn 首次提出应用模糊数学判据的 ISODATA 集群算法——模糊 ISODATA。算法通过每个样本点对各类的隶属度矩阵表示分类结果,通过不断修改聚类中心的位置来进行分类。1976 年 J. C. Bezdek 把 Dunn 的方法推广到更一般的情形,并得到了一些有益的结论,其中包括新的判据,隶属度函数与聚类中心的计算公式。J. C. Bezdek 于 1979 年用 W. Zangwill 的理论证明了模糊 ISODATA 的收敛性。该方法已在行星跟踪系统、心脏病分析和天气预报等方面得到了应用。

5.8.2 模糊 ISODATA 算法的基本原理

J. C. Bezdek 在普通分类基础上,利用模糊集合的概念提出了模糊分类问题。认为被分类对象集合 $\boldsymbol{X}$ 中的样本 $\boldsymbol{X}_i$ 以一定的隶属度属于某一类,即所有的样本都分别以不同的隶属度属于某一类。因此每一类就被认为是样本集 $\boldsymbol{X}$ 上的一个模糊子集,于是,每一种这样的分类结果所对应的分类矩阵就是一个模糊矩阵。模糊 ISODATA 聚类方法从选择的初始聚类中心出发,根据目标函数,用数学迭代计算的方法反复修改模糊矩阵和聚类中心,并对类别进行合并、分解和删除等操作,直到合理为止。

设有限样本集(论域) $X=\{X_1,X_2,\cdots,X_N\}$,每一个样本有 s 个特征 $X_j=\{x_{j1},x_{j2},\cdots,x_{js}\}$,$(j=1,2,\cdots,N)$,即欲把样本的特征矩阵

$$\boldsymbol{X}_{N\times s}=\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}=\begin{bmatrix}x_{11}&x_{12}&\cdots&x_{1s}\\x_{21}&x_{22}&\cdots&x_{2s}\\\vdots&\vdots&\ddots&\vdots\\x_{N1}&x_{N2}&\cdots&x_{Ns}\end{bmatrix}\tag{5-43}$$

分为 K 类($2\leqslant K\leqslant N$),则 N 个样本划分为 K 类的模糊分类矩阵为

$$\boldsymbol{U}_{K\times N}=\begin{bmatrix}\mu_1\\\mu_2\\\vdots\\\mu_K\end{bmatrix}=\begin{bmatrix}\mu_{11}&\mu_{12}&\cdots&\mu_{1N}\\\mu_{21}&\mu_{22}&\cdots&\mu_{2N}\\\vdots&\vdots&\ddots&\vdots\\\mu_{K1}&\mu_{K2}&\cdots&\mu_{KN}\end{bmatrix}\tag{5-44}$$

其满足下列三个条件：

(1) $0\leqslant\mu_{ij}\leqslant 1,\quad i=1,2,\cdots,K;\ j=1,2,\cdots,N$ (5-45)

(2) $\sum\limits_{i=1}^{N}\mu_{ij}=1,\quad j=1,2,\cdots,N$ (5-46)

(3) $0<\sum\limits_{i=1}^{N}\mu_{ij}<N,\quad i=1,2,\cdots,K$ (5-47)

条件(2)表明每一样本属于各类的隶属度之和为1；条件(3)表明每一类模糊集不可能是空集合,即总有样本不同程度的隶属于某类。

定义 K 个聚类中心 $Z=\{Z_1,Z_2,\cdots,Z_K\}$。其中,$Z_i=\{Z_{i1},Z_{i2},\cdots,Z_{is}\}$,$i=1,2,\cdots,K$。

$$\boldsymbol{Z}_{K\times s}=\begin{bmatrix}Z_1\\Z_2\\\vdots\\Z_K\end{bmatrix}=\begin{bmatrix}z_{11}&z_{12}&\cdots&z_{1s}\\z_{21}&z_{22}&\cdots&z_{2s}\\\vdots&\vdots&\ddots&\vdots\\z_{K1}&z_{K2}&\cdots&z_{Ks}\end{bmatrix}\tag{5-48}$$

第 i 类的中心 Z_i 即人为假想的理想样本,它对应的 s 个指标值是该类样本所对应的指标值的平均值

$$Z_{ij}=\frac{\sum\limits_{k=1}^{N}(\mu_{ik})^mX_{kj}}{\sum\limits_{k=1}^{N}(\mu_{ik})^m},\quad i=1,2,\cdots,K;\ j=1,2,\cdots,s\tag{5-49}$$

构造准则函数

$$J=\sum_{i=1}^{K}\sum_{j=1}^{N}[\mu_{ij}(L+1)]^m\|X_j-Z_i\|^2\tag{5-50}$$

其中,$\|X_j-Z_i\|$ 表示第 j 个样本与第 i 类中心之间的欧氏距离；J 表示所有待聚类样本与所属类的聚类中心之间距离的平方和。

为了确定最佳分类结果,就是寻求最佳划分矩阵 $\boldsymbol{U}$ 和对应的聚类中心 $\boldsymbol{Z}$,使 J 达到极小。Dunn 证明了求上述泛函的极小值的问题可解。

5.8.3 模糊 ISODATA 算法的基本步骤

(1) 选择初始聚类中心 $Z_i(0)$。例如,可以将全体样本的均值作为第一个聚类中心,然后在每个特征方向上加和减一个均方差,共得$(2n+1)$个聚类中心,n 是样本的维数(特征

数)。也可以用其他方法选择初始聚类中心。

(2) 若已选择了 K 个初始聚类中心,接着利用模糊 K 均值算法对样本进行聚类。由于现在得到的不是初始隶属度矩阵 $\boldsymbol{U}(0)$,而是各类聚类中心,所以算法应从模糊 K 均值算法的第四步开始,即直接计算下一步的隶属度矩阵 $\boldsymbol{U}(0)$。继续 K 均值算法直到收敛为止,最终得到隶属度矩阵 $\boldsymbol{U}$ 和 K 个聚类中心 $\boldsymbol{Z}=\{Z_1,Z_2,\cdots,Z_K\}$。然后进行类别调整。

① 计算初始隶属度矩阵 $\boldsymbol{U}(0)$,矩阵元素的计算方法为

$$\mu_{ij}(0)=\frac{1}{\sum_{p=1}^{K}\left(\frac{d_{ij}}{d_{pj}}\right)^{\frac{2}{(m-1)}}},\quad i=1,2,\cdots,K;\ j=1,2,\cdots,N;\ m\geqslant 2 \tag{5-51}$$

式中,d_{ij} 是第 j 个样本到第 i 类初始聚类中心 $Z_i(0)$ 的距离。为避免分母为 0,特规定:若 $d_{ij}=0$,则 $\mu_{ij}=1$,$\mu_{pj}(0)=0(p\neq i)$。可见,d_{ij} 越大,$\mu_{ij}(0)$ 越小。

② 求各类的新的聚类中心 $Z_i(L)$,L 为迭代次数。

$$\boldsymbol{Z}_i(L)=\frac{\sum_{j=1}^{N}[\mu_{ij}(L)]^m X_j}{\sum_{j=1}^{N}[\mu_{ij}(L)]^m},\quad j=1,2,\cdots,K \tag{5-52}$$

式中,参数 $m\geqslant 2$,是一个控制聚类结果模糊程度的常数。可以看出各聚类中心的计算必须用到全部的 N 个样本,这是与非模糊的 K 均值算法的区别之一。在 K 均值算法中,某一类的聚类中心仅由该类样本决定,不涉及其他类。

③ 计算新的隶属度矩阵 $\boldsymbol{U}(L+1)$,矩阵元素的计算方法为

$$\mu_{ij}(L+1)=\frac{1}{\sum_{p=1}^{K}\left(\frac{d_{ij}}{d_{pj}}\right)^{\frac{2}{(m-1)}}},\quad i=1,2,\cdots,K;\ j=1,2,\cdots,N;\ m\geqslant 2 \tag{5-53}$$

式中,d_{ij} 是第 L 次迭代完成时,第 j 个样本到第 i 类聚类中心 $Z_i(L)$ 的距离。为避免分母为 0,特规定:若 $d_{ij}=0$,则 $\mu_{ij}(L+1)=1$,$\mu_{pj}(L+1)=0(p\neq i)$。可见,d_{ij} 越大,$\mu_{ij}(L+1)$ 越小。

④ 回到第③步,重复至收敛。收敛条件为 $\max\limits_{i,j}\{|\mu_{ij}(L+1)-\mu_{ij}(L)|\}\leqslant\varepsilon$,其中,$\varepsilon$ 为规定的参数。

(3) 类别调整。调整分三种情形:

① 合并。

假定各聚类中心之间的平均距离为 D,则取合并阈值为

$$M_{\text{ind}}=D[1-F(K)] \tag{5-54}$$

其中,$F(K)$ 是人为构造的函数,$0\leqslant F(K)\leqslant 1$,而且 $F(K)$ 应是 K 的减函数,通常取 $F(K)=\frac{1}{K^\alpha}$,α 是一个可选择的参数。可见,若 D 确定,则 K 越大时 M_{ind} 也越大,即合并越容易发生。

若聚类中心 Z_i 和 Z_j 间的距离小于 M_{ind},则合并这两个点而得到新的聚类中心 Z_L,Z_L 为

$$Z_L=\frac{\left(\sum_{p=1}^{N}\mu_{ip}\right)Z_i+\left(\sum_{p=1}^{N}\mu_{ip}\right)Z_j}{\sum_{p=1}^{N}\mu_{ip}+\sum_{p=1}^{N}\mu_{ip}} \tag{5-55}$$

式中，N 为样本个数。可见，Z_L 是 Z_i 和 Z_j 的加权平均，而所用的权系数便是全体样本对 ω_i 和 ω_j 两类的隶属度。

② 分解。

首先计算各类在每个特征方向上的“模糊化方差”。对于 ω_i 类的第 j 个特征，模糊化方差的计算公式为

$$S_{ij}^2 = \frac{1}{N+1}\sum_{p=1}^{N}\mu_{ip^{\beta}}(x_{pj}-z_{ij})^2,\quad j=1,2,\cdots,n;\ i=1,2,\cdots,K \tag{5-56}$$

式中 β 是参数，通常选 $\beta=1$。x_{pj}，z_{ij} 分别表示样本 X_p 和聚类中心 Z_i 的第 j 个特征值。$S_{ij}=\sqrt{S_{ij}^2}$，全体 S_{ij} 的平均值记作 S，然后求阈值

$$F_{\text{std}} = S[1+G(K)] \tag{5-57}$$

$G(K)$ 是类数 K 的增函数，通常取 $G(K)=K^{\gamma}$，γ 是参数。式(5-57)表明，当 S 确定时，类数 K 越大，越不易分解。下面分两步进行分解：

第一步，检查各类的“聚集程度”。对于任一类 ω_i，取 $\text{sum}_i=\sum_{p=1}^{N}t_{ip}\mu_{ip}$，其中，$t_{ip}=\begin{cases}0, \mu_{ip}\leqslant\theta\\1, \mu_{ip}>\theta\end{cases}$。然后取 $T_i=\sum_{p=1}^{N}t_{ip}$，$C_i=\text{sum}_i/T_i$，其中，θ 为一参数，$0<\theta<0.5$。C_i 表示 ω_i 类的聚集程度。上两式的含义是对于每一类 ω_i，首先舍去那些对它的隶属度太小的样本，然后计算其他各样本对该类的平均隶属度 C_i。若 $C_i>A_{\text{vms}}$（A_{vms}为参数），则表示 ω_i 类的聚集程度较高，不必进行分解；否则考虑下一步。

第二步，分解。对于任一不满足 $C_i>A_{\text{vms}}$的 ω_i 类考虑其每个 S_{ij}，若 $S_{ij}>F_{\text{std}}$，便在第 j 个特征方向上对聚类中心 Z_i 加和减 kS_{ij}（k 为分裂系数，$0<k\leqslant1$），得到两个新的聚类中心。

注意，这里每个量的计算都考虑到了全体样本对各类的隶属度。

③ 删除。

删除某个类 ω_i 或聚类中心 Z_i 的条件有两个。

条件1：$T_i\leqslant\delta N/K$，δ 是参数，T_i 见上式，它表示对 ω_i 类隶属度超过 θ 的点数。这一条件表示对 ω_i 类隶属度高的点很少，应该删除。

条件2：$C_i\leqslant A_{\text{vms}}$，但 ω_i 类不满足分解条件，即对所有的 j，$S_{ij}\leqslant F_{\text{std}}$。这个条件表明，在 Z_i 的周围存在着一批样本点，它们的聚集程度不高，但也不是非常分散。这时，我们认为 Z_i 也不是一个理想的聚类中心。

符合以上两个条件之一者，将被删除。

如果在第(3)步类别调整中进行了合并、分解或删除，则在每次处理后都应进行下面所指出的讨论，并在全部处理结束后做出一个选择：停止在某个结果上，或者转到第(2)步重新迭代。如果在第(3)步中没有进行任何类别调整，则表示已经不需要改进结果，计算停止。

(4) 关于最佳类数或最佳结果的讨论。

上述所得为预选定分类数 K 时的最优解，为局部最优解。最优聚类数 K 可借助下列判定聚类效果的指标值得到。

分类系数：$F(R)=\frac{1}{n}\sum_{i=1}^{K}\sum_{j=1}^{N}\mu_{ij}^2$，$F$ 越接近1，聚类效果越好。

平均模糊熵：$H(R)=\frac{1}{n}\sum_{i=1}^{K}\sum_{j=1}^{N}\mu_{ij}\ln(\mu_{ij})$，$H$ 越接近于 0，聚类效果越好。

由此，可以分别选定 $K(2<K\leqslant N)$，计算其所得聚类结果的聚类指标值并进行比较，求得最优聚类个数 K，即满足 F 最接近 1 或 H 最接近 0 的 K 值。

(5) 分类清晰化。有两种方法：

① X_j 与哪一类的聚类中心最接近，就将 X_j 归到哪一类。即：$\forall X_j\in X$，若 $\|X_j-Z_{\omega_i}\|=\min\|X_j-Z_{\omega_i}\|$，就将 $X_j\in\omega_i$ 类。

② X_j 对哪一类的隶属度最大，就将它归于哪一类。即：在 U 的第 j 列中，若 $\mu_{ij}(L+1)=\max\limits_{1\leqslant p\leqslant K}\mu_{pj}(L+1)(j=1,2,\cdots,N)$，则 $X_j\in\omega_i$ 类。

当算法结束时，就得到了各类的聚类中心以及表示各样本对各类隶属程度的隶属度矩阵，模糊聚类到此结束。这时，准则函数 $J=\sum_{i=1}^{K}\sum_{j=1}^{N}[\mu_{ij}(L+1)]^m\|X_j-Z_i\|^2$ 达到最小。

5.8.4 模糊 ISODATA 算法的 MATLAB 程序实现

这里使用表 1-2 的 59 组数据为例来实现聚类，模糊 ISODATA 算法的 MATLAB 程序流程图如图 5-31 所示。

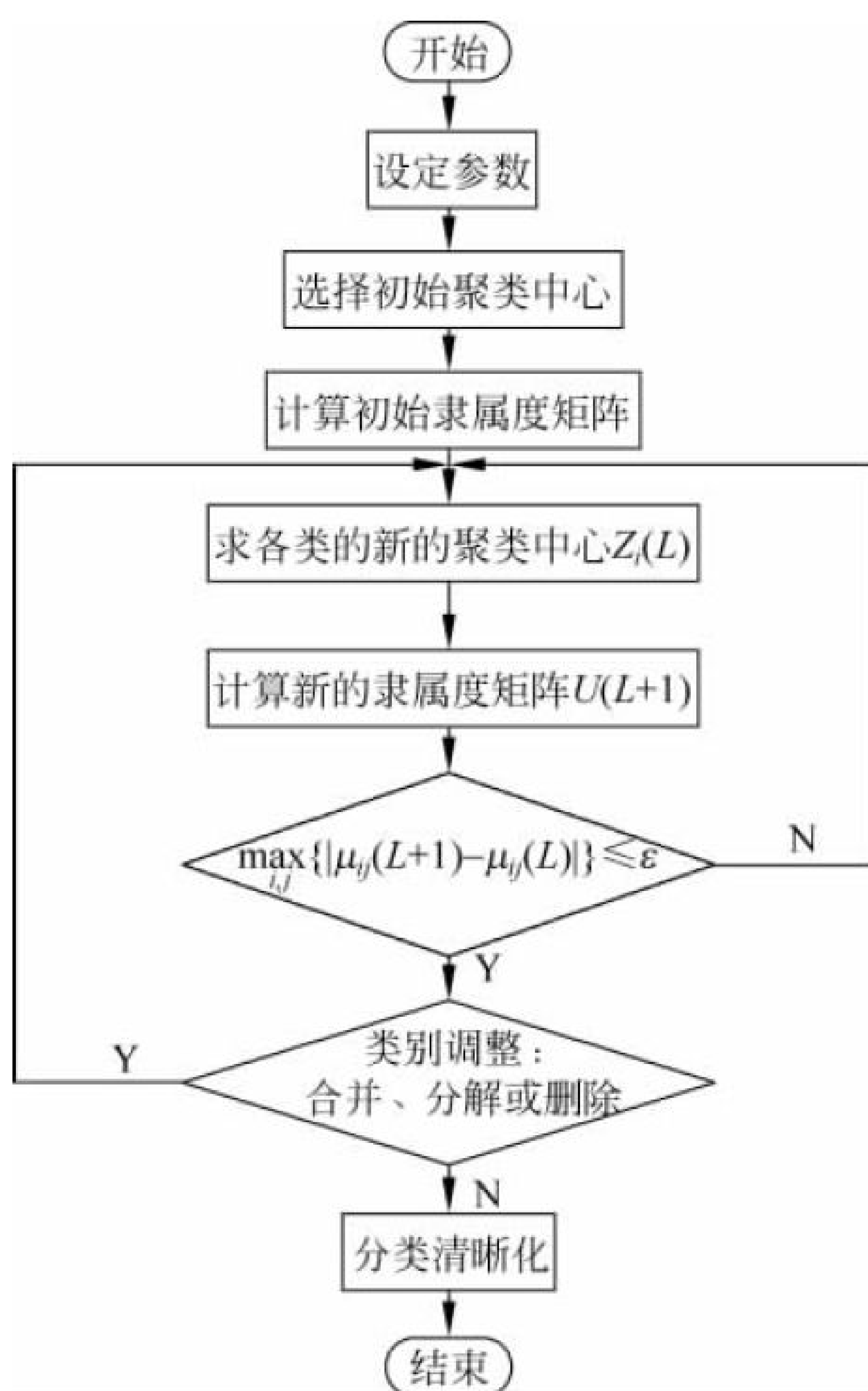


图 5-31 模糊 ISODATA 算法流程图

1. 调节参数初始化

调节参数初始化代码如下：

```
Nc = 4;                  % 初始聚类中心数目
m = 2;                   % 控制聚类结果模糊程度
L = 0;                   % 迭代次数
Lmax = 1000;             % 最大迭代次数
Nc_all = ones(Lmax,2);   % 各次迭代的分类数
Udmax = 10;              %最后一次隶属度与前一次隶属度差值的初始值
e = 0.00005;             % 收敛参数
a = 0.33;                % 合并阈值系数
b = 1;                   % 模糊化方差参数(通常取 1)
r = 0.1;                 % 分解阈值参数(算法使用者掌握的参数,控制 G(K)的上升速度)
f = 0.68;                % 隶属度阈值(一般取值 0～0.5)
Avms = 0.83;             % 平均隶属度阈值(一般应大于 0.5,在 0.55～0.6 取值比较适宜)
k_divide = 0.9;          %分裂 1 数(取 0～1)
w = 0.2;                 % 删除条件参数
```

2. 模糊 ISODATA 函数

模糊 ISODATA 的函数定义代码如下：

```
function [X,Z,U,Nc,L,Dcc,Dccm,Mind, S,Smean,Fstd,T,C,k_delete,Dpc] =
FussyISODATA_function(data,Nc,m,L,Lmax,Nc_all,Udmax,e,a,b,r,f,Avms,k_divide,w)
%       data     样本特征库
%         Nc     初始聚类中心数目
%          m     控制聚类结果模糊程度
%          L     迭代次数
%       Lmax     最大迭代次数
%     Nc_all     各次迭代的分类数
%      Udmax     最后一次隶属度与前一次隶属度差值的初始值
%          e     收敛参数
%          a     合并阈值系数
%          b     模糊化方差参数
%          r     分解阈值参数
%          f     隶属度阈值
%       Avms     平均隶属度阈值
%   k_divide     分裂系数
%          w     删除条件参数
% 返回值:
%          X     样本结构体数组: 样本特征、所属类别
%          Z     聚类中心结构体数组: 聚类中心特征、所属类别及其包含的样本数
%          U     隶属度矩阵
%         Nc     聚类中心数目
%          L     迭代次数
%        Dcc     两两聚类中心之间的距离矩阵
%       Dccm     两两聚类中心之间的距离的平均值
%       Mind     合并阈值
%          S     各类在每个特征方向上的模糊化标准差矩阵
%      Smean     模糊化标准差平均值
%       Fstd     分解阈值
%          T     各类超过隶属度阈值 f 的样本数矩阵
%          C     各类的聚集程度矩阵
%   k_delete     删除阈值
%        Dpc     各样本点到各聚类中心的距离矩阵
```

3. 聚类函数

聚类函数的代码如下：

```
function [Z,U,Nc,Nc_all,L,Dpc] = FussyISODATA_newcentre(X,Z,U,Nc,Nc_all,Np,Nq,e,m,L,Lmax,
Udmax)
% 名称:      FussyISODATA_newcentre
% 参数:
%          X   样本结构体数组: 样本特征、所属类别
%          Z   聚类中心结构体数组: 聚类中心特征、所属类别及其包含的样本数
%          U   隶属度矩阵
%         Nc   聚类中心数目
%     Nc_all   各次迭代的分类数
%         Np   样本数目
%         Nq   样本维数
%          e   收敛参数
%          m   控制聚类结果模糊程度
%          L   迭代次数
%       Lmax   最大迭代次数
%      Udmax   最后一次隶属度与前一次隶属度差值的初始值
% 返回值:
%          Z   聚类中心结构体数组: 聚类中心特征、所属类别及其包含的样本数
%          U   隶属度矩阵
%         Nc   聚类中心数目
%     Nc_all   各次迭代的分类数
%          L   迭代次数
%        Dpc   各样本点到各聚类中心的距离矩阵
% 功能:
%              重复计算新的隶属度矩阵及聚类中心,直至收敛
```

4. 类别调整函数

类别调整函数代码如下：

```
function [Z,U,Nc,Dcc,Dccm,Mind, S,Smean,Fstd,T,C,k_delete] = FussyISODATA_adjust(X,Z,U,Nc,
Np,Nq,a,f,Avms,b,r,k_divide,w)
% 名称:      FussyISODATA_adjust
% 参数:
%            X   样本结构体数组: 样本特征、所属类别
%            Z   聚类中心结构体数组: 聚类中心特征、所属类别及其包含的样本数
%            U   隶属度矩阵
%           Nc   聚类中心数目
%           Np   样本数目
%           Nq   样本维数
%            a   合并阈值系数
%            f   隶属度阈值
%         Avms   平均隶属度阈值
%            b   模糊化方差参数
```

```
%          r       分解阈值参数
%  k_divide        分裂系数
%          w       删除条件参数
% 返回值:
%          Z       聚类中心结构体数组:聚类中心特征、所属类别及其包含的样本数
%          U       隶属度矩阵
%         Nc       聚类中心数目
%        Dcc       两两聚类中心之间的距离矩阵
%       Dccm       两两聚类中心之间的距离的平均值
%       Mind       合并阈值
%          S       各类在每个特征方向上的模糊化标准差矩阵
%      Smean       模糊化标准差平均值
%       Fstd       分解阈值
%          T       各类超过隶属度阈值 f 的样本数矩阵
%          C       各类的聚集程度矩阵
%  k_delete        删除阈值
% 功能:
%                  调整聚类结果:合并、分解或者删除
```

5. 完整 MATLAB 程序及仿真结果

完整的 MATLAB 程序代码如下:

```
close all;                                   %关闭窗口
clear all;                                   %清空工作空间
data = xlsread('Nfsensor.csv','Nfsensor');%读入样本数据
Nc = 4;                                      %初始聚类中心数目
m = 2;                                       %控制聚类结果模糊程度
L = 0;                                       %迭代次数
Lmax = 1000;                                 %最大迭代次数
Nc_all = ones(Lmax,2);                       %各次迭代的分类数
Udmax = 10;                                  %最后一次隶属度与前一次隶属度差值的初始值
e = 0.00005;                                 %收敛参数
a = 0.33;                                    %合并阈值系数
b = 1;                                       %模糊化方差参数(通常取 1)
r = 0.1;                    %分解阈值参数(算法使用者掌握的参数,控制 G(K)的上升速度)
f = 0.68;                                    %隶属度阈值(一般取值 0~0.5)
Avms = 0.83;                %平均隶属度阈值(一般应大于 0.5,在 0.55~0.6 取值比较适宜)
k_divide = 0.9;                              %分裂 1 数(取 0~1)
w = 0.2;                                     %删除条件参数
Nc_start = Nc;
%调用 Fuzzy ISODATA 函数
[X,Z,U,Nc,L,Dcc,Dccm,Mind, S,Smean,Fstd,T,C,k_delete,Dpc] = FussyISODATA_function(data,
Nc,m,L,Lmax,Nc_all,Udmax,e,a,b,r,f,Avms,k_divide,w)
[Np,Nq] = size(data);                        %Np 样本数目; Nq 样本维数
%将聚类结果在三维图中显示
figure;
```

```
hold on;
for i = 1:Np
    for j = 1:Nc
      if Nc > 8
          disp('聚类中心数目大于 8 个');
      else
        switch X(i,1).category
          case 1
            plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'b*');
                                                        %第 1 类样本,蓝色 *
            grid on;box;
            plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                        %第 1 类聚类中心,黑色 o
            grid on;
          case 2
            plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'gd');
                                                        %第 2 类样本,绿色菱形
            grid on;
            plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                        %第 2 类聚类中心,黑色 o
            grid on;
          case 3
            plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'rs');
                                                        %第 3 类样本,红色方块
            grid on;
            plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                        %第 3 类聚类中心,黑色 o
            grid on;
          case 4
            plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'c+');
                                                        %第 4 类样本,青色 +
            grid on;
            plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                        %第 4 类聚类中心,黑色 o
            grid on;
          case 5
            plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'mx');
                                                        %第 5 类样本,品红色 x
            grid on;
            plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                        %第 5 类聚类中心,黑色 o
            grid on;
          case 6
            plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'yh');
                                                        %第 6 类样本,黄色六角星
            grid on;
            plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                        %第 6 类聚类中心,黑色 o
            grid on;
```

```
            case 7
              plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'k.');
                                                    %第 7 类样本,黑色
              grid on;
              plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                    %第 7 类聚类中心,黑色 o
              grid on;
            case 8
              plot3(X(i,1).feature(1,1),X(i,1).feature(1,2),X(i,1).feature(1,3),'rp');
                                                    %第 8 类样本,红色五角星
              grid on;
              plot3(Z(j,1).feature(1,1),Z(j,1).feature(1,2),Z(j,1).feature(1,3),'ko');
                                                    %第 8 类聚类中心,黑色 o
              grid on;
        end
      end
    end
end
%显示方向轴名称
xlabel('第一特征');
ylabel('第二特征');
zlabel('第三特征');
title('程序运行结果');
%显示各聚类中心
for i = 1:Nc
    A(i,:) = Z(i,1).feature(1,:);
end
%显示各样本所属类别
for i = 1:Np
    B(i,1) = X(i,1).category;
End
function [X,Z,U,Nc,L,Dcc,Dccm,Mind, S,Smean,Fstd,T,C,k_delete,Dpc] =
FussyISODATA_function(data,Nc,m,L,Lmax,Nc_all,Udmax,e,a,b,r,f,Avms,k_divide,w)
    Ln = zeros(Lmax,1);
    [Np,Nq] = size(data);                     %Np 为样本数目; Nq 为样本维数
    for i = 1:Np
      X(i,1).feature = [data(i,:)];           %将样本数据导入样本结构体数组
    end
    %选取 Nc 个初始聚类中心
    for i = 1:Nc                              %选取前 Nc 个样本为初始聚类中心
      X(i,1).category = i;                    %第 i 个样本所属类别
      Z(i,1).feature = X(i,1).feature;        %选取初始聚类中心
      Z(i,1).index = i;                       %第 i 聚类
      Z(i,1).patternNum = 1;                  %第 i 聚类中样本数
    end
    %计算所有样本到各初始聚类中心的距离
    Dpc = zeros(Nc,Np);
    for i = 1:Nc
      for j = 1:Np
```

```
            Dpc(i,j) = sqrt((X(j,1).feature(1,1) - Z(i,1).feature(1,1))^2 + (X(j,1)
.feature(1,2) - Z(i,1).feature(1,2))^2 + (X(j,1).feature(1,3) - Z(i,1).feature(1,3))^2);
        end
    end
    %计算初始隶属度矩阵 U(0)
    for i = 1:Nc
        for j = 1:Np
            if Dpc(i,j) == 0                        %Dpc(i,j) = 0 时,U(i,j) = 1
                U(i,j) = 1;
            else
                d = 0;
                for k = 1:Nc
                    if (Dpc(k,j) == 0) & (k~ = i)     %Dpc(i,j) = 0 且 k~ = i 时,U(i,j) = 0
                        U(k,j) = 0;
                    elseif (Dpc(k,j) == 0) & (k == i)  %Dpc(i,j) = 0 且 k = i 时,U(i,j) = 1
                        U(k,j) = 1;
                    else
                        d = d + (Dpc(i,j)/Dpc(k,j))^(2/(m-1));  %Dpc(i,j)~ = 0 时,计算隶
                                                                %属度函数的分母
                    end
                end
                U(i,j) = 1/d;                       %计算隶属度
            end
        end
    end
    %调用求新的聚类中心及隶属度矩阵的函数
    [Z,U,Nc,Nc_all,L,Dpc] = FussyISODATA_newcentre(X,Z,U,Nc,Nc_all,Np,Nq,e,m,L,Lmax,
Udmax)
    %调用类别调整函数,对聚类结果进行合并、分解或者删除
    [Z,U,Nc,Dcc,Dccm,Mind, S,Smean,Fstd,T,C,k_delete] = FussyISODATA_adjust(X,Z,U,Nc,
Np,Nq,a,f,Avms,b,r,k_divide,w)
    %类别调整后,重新计算所有样本到各新聚类中心的距离
    Dpc = zeros(Nc,Np);
    for i = 1:Nc
        for j = 1:Np
            Dpc(i,j) = sqrt((X(j,1).feature(1,1) - Z(i,1).feature(1,1))^2 + (X(j,1)
.feature(1,2) - Z(i,1).feature(1,2))^2 + (X(j,1).feature(1,3) - Z(i,1).feature(1,3))^2);
        end
    end
    %类别调整后,计算新隶属度矩阵
    U = zeros(Nc,Np);
    for i = 1:Nc
        for j = 1:Np
            if Dpc(i,j) == 0                        %Dpc(i,j) = 0 时,U(i,j) = 1
                U(i,j) = 1;
            else
                d = 0;
                for k = 1:Nc
                    if (Dpc(k,j) == 0) & (k~ = i)
```

```
                                        % Dpc(i,j) = 0 且 k~ = i 时,U(i,j) = 0
                        U(i,j) = 1;
                    elseif (Dpc(k,j) == 0) & (k == i)
                                        % Dpc(i,j) = 0 且 k = i 时,U(i,j) = 1
                        U(k,j) = 1;
                    else
                        d = d + (Dpc(i,j)/Dpc(k,j))^(2/(m-1));
                                        % Dpc(i,j)~ = 0 时,计算隶属度函数的分母
                    end
                end
                U(i,j) = 1/d;           % 计算隶属度
            end
        end
    end
    % 类别调整后,调用求新的聚类中心及隶属度矩阵的函数,重新计算聚类中心
    [Z,U,Nc,Nc_all,L,Dpc] = FussyISODATA_newcentre(X,Z,U,Nc,Nc_all,Np,Nq,e,m,L,
Lmax,Udmax)
    % 重新划分样本类别
    for i = 1:Np
        Umax(1,i) = max(U(:,i));      % 找出各样本对所有聚类中心隶属度的最大值
    end
    for i = 1:Nc
        Z(i,1).patternNum = 0;        % 初始化各类包含的样本数
    end
    for i = 1:Np
        [i1,i2] = find(U(:,i) == Umax(1,i));
                    % 找出各样本对所有聚类中心隶属度的最大值在隶属度矩阵中的位置
        if size(i1) == 1
                    % 各样本对所有聚类中心隶属度的最大值只有 1 个
            X(i,1).category = i1;
                    % 第 i 个样本所属的类别
        else
                    % 各样本对所有聚类中心隶属度的最大值不只 1 个
            i1 = i1(fix(rand * size(i1) + 1));
                    % 从多个隶属度相同的聚类中心中,随机选取一类
            X(i,1).category = i1;
                    % 第 i 个样本所属的类别
        end
    end
  function [Z,U,Nc,Nc_all,L,Dpc] = FussyISODATA_newcentre(X,Z,U,Nc,Nc_all,Np,Nq,e,
m,L,Lmax,Udmax)
    while Udmax > e
                    % 重复计算新的聚类中心和隶属度矩阵,至满足收敛条件
      % 判断是否超过最大迭代次数,超过则跳出子函数
      if L > Lmax
            return;
      end
        Dpc = zeros(Nc,Np);           % 初始化各样本点到各聚类中心的距离矩阵
      % 计算新的聚类中心
      U1 = U.^m;                      % 求隶属度矩阵各值的 m 次方
```

```
            A = zeros(1,Nq);                    % 定义一个中间变量,全零矩阵
            B = sum((U.^m)');                   % 求隶属度矩阵各值的 m 次方后,各行的和
            for i = 1:Nc
                for j = 1:Np
                    A(1,:) = A(1,:) + U1(i,j) * X(j).feature(1,:);
                                                % 求聚类中心函数的分子
                end
                Z(i,1).feature(1,:) = A(1,:)./B(1,i);
                                                % 求新的聚类中心
                A = zeros(1,Nq);
              end
            Up = U;                             % Up 为第 L 次隶属度矩阵
             % 计算所有样本到各聚类中心的距离
            for i = 1:Nc
                for j = 1:Np
                    Dpc(i,j) = sqrt((X(j,1).feature(1,1) - Z(i,1).feature(1,1))^2 + (X(j,
1).feature(1,2) - Z(i,1).feature(1,2))^2 + (X(j,1).feature(1,3) - Z(i,1).feature(1,3))^2);
                end
            end
             % 计算第 L+1 次隶属度矩阵 U(L+1)
            for i = 1:Nc
                for j = 1:Np
                    if Dpc(i,j) == 0
                        U(i,j) = 1;             % U 为第 L+1 次隶属度矩阵
                    else
                        d = 0;
                        for k = 1:Nc
                            if (Dpc(k,j) == 0) & (k~=i)
                                U(k,j) = 0;
                            elseif (Dpc(k,j) == 0) & (k==i)
                                U(k,j) = 1;
                            else
                                d = d + (Dpc(i,j)/Dpc(k,j))^(2/(m-1));
                            end
                        end
                        U(i,j) = 1/d;
                    end
                end
            end
            Udmax = max(max(U-Up));             % 计算收敛条件值
            L = L + 1;                          % 迭代次数+1
            Nc_all(L,1) = Nc;                   % 记录第 L 次迭代的聚类中心数
          end
        function [Z,U,Nc,Dcc,Dccm,Mind, S,Smean,Fstd,T,C,k_delete] =
FussyISODATA_adjust(X,Z,U,Nc,Np,Nq,a,f,Avms,b,r,k_divide,w)
          % 变量初始化
          Dcc = zeros(Nc,Nc);                   % 两两聚类中心之间的距离矩阵
          Dccm = 0;                             % 两两聚类中心之间的距离的平均值
          Mind = 0;                             % 合并阈值
          S = zeros(Nc,Nq);                     % 各类在每个特征方向上的模糊化标准差矩阵
```

```
        Smean = 0;                                    %模糊化标准差平均值
        Fstd = 0;                                     %分解阈值
        T = zeros(Nc,1);                              %各类超过隶属度阈值 f 的样本数矩阵
        C = zeros(Nc,1);                              %各类的聚集程度矩阵
        k_delete = 0;                                 %删除阈值
        %1. 合并
        %计算各聚类中心之间的距离 Dcc(i,j)
        DccSum = 0;                                   %所有聚类中心距离的和
        for i = 1:(Nc-1)
            for j = (i+1):Nc
                %两两聚类中心之间的距离
                Dcc(i,j) = sqrt((Z(j,1).feature(1,1) - Z(i,1).feature(1,1))^2 + ((Z(j,1)
.feature(1,2) - Z(i,1).feature(1,2))^2 + ((Z(j,1).feature(1,3) - Z(i,1).feature(1,3))^2)));
                DccSum = DccSum + Dcc(i,j);           %所有聚类中心距离的和
            end
        end
        %计算各聚类中心之间的平均距离 Dccm
        Ncc = nchoosek(Nc,2);                         %两两聚类中心的组合数
        Dccm = DccSum/Ncc;                            %两两聚类中心之间的距离的平均值
        %计算合并阈值
        Mind = Dccm*(1-1/(Nc^a));
        %根据合并阈值判断,合并聚类中心,得到新的聚类中心
        Y1 = Z;                                       %中间变量
        flag1 = 0;                                    %中间标志
        Nc_combine = Nc;                              %合并后的聚类中心数
        N_combine = 0;                                %合并次数
        for i = 1:(Nc-1)
            for j = (i+1):Nc
                if Dcc(i,j) < Mind
                            %两聚类中心之间的距离小于合并阈值时,合并这两个聚类中心
                    ki = sum(U(i,:));
                    kj = sum(U(j,:));
                    Y1(i).feature(1,:) = (ki*Z(i,1).feature(1,:) + kj*Z(j,1)
.feature(1,:))/(ki + kj);                             %合并后的聚类中心
                    Y1(j).feature(1,:) = [zeros(1,Nq)];
                                                      %被合并的聚类中心赋 0
                    N_combine = N_combine + 1;        %合并次数+1
                    Nc_combine = Nc_combine - 1;      %类别数-1
                  if (Nc_combine <= 2) | (Nc_combine >= 8)
                                                      %分类数不满足要求时,跳出循环
                        flag1 = 1;
                            Z = Y1;
                        break;
                    end
                end
            end
            if flag1 == 1
                break;
            end
        end
```

```
        %2. 分解
        %计算模糊化方差
        S_mid = 0;                                  %中间变量
        for i = 1:Nc
            for j = 1:Nq
                for p = 1:Np
                    S_mid = S_mid + (U(i,p)^b) * ((X(p,1).feature(1,j) - Z(i,1)
.feature(1,j))^2);
                                                    %模糊化方差的分子
            end
            S2(i,j) = S_mid/(Np-1);                 %模糊化方差
            S(i,j) = sqrt(S2(i,j));                 %模糊化标准差
        end
    end
    %计算全体模糊化方差的平均值
    Smean = sum(sum(S))/(Nq*Nc);
    %计算分解阈值
    Fstd = Smean*(Nc^r);
    %检查各类的聚集程度
    Sum = zeros(Nc,1);                              %聚集程度 C 的分子
    for i = 1:Nc
        for p = 1:Np
            if U(i,p)>f
                t(i,p) = 1;
            else
                t(i,p) = 0;
            end
            T(i,1) = T(i,1) + t(i,p);               %计算聚集程度 C 的分母
            Sum(i,1) = Sum(i,1) + t(i,p)*U(i,p);    %计算聚集程度 C 的分子
        end
    end
    C = Sum./T;                                     %计算聚集程度矩阵
    %根据平均分解阈值判断是否进行分解
    Nc_divide = Nc;                                 %分解后的聚类中心数
    N_divide = 0;                                   %分解次数
    flag2 = 0;                                      %中间标志
    Y2 = Z;
    for i = 1:Nc
        if C(i,1)<=Avms
            for j = 1:3
                if S(i,j)>Fstd
                    N_divide = N_divide+1;          %分解次数+1
                    Zdiv1 = Z(i,1).feature(1,:);
                    Zdiv2 = Z(i,1).feature(1,:);
                    Zdiv1(i,j) = Z(i,1).feature(1,j) + k_divide*S(i,j);
                                                    %分解后,新的聚类中心 1
                    Zdiv2(i,j) = Z(i,1).feature(1,j) - k_divide*S(i,j);
                                                    %分解后,新的聚类中心 2
                     Y2(i,1).feature(1,:) = [Zdiv1(1,:)];
                                        %分解后,新的聚类中心 1 写入聚类中心结构体数组第 i 项
                     Y2(Nc+N_divide,1).feature(1,:) = [Zdiv2(1,:)];
                      %分解后,新的聚类中心 2 写入聚类中心结构体数组第 Nc+N_divide 项
                   Nc_divide = Nc_divide+1;         %分解后的聚类中心数
```

```
                if (Nc_divide <= 2) | (Nc_divide >= 8)  % 分类数不满足要求时,跳出循环
                    flag2 = 1;  % 中间标志为 1,跳出循环
                    break;
                end
            end
        end
    end
if flag2 == 1;  % 中间标志为 1,跳出循环
        break;
    end
end
%3. 删除
Nc_delete = Nc;                                      % 删除后的聚类中心数
N_delete = 0;                                        % 删除次数
flag3 = 0;                                           % 中间标志
Y3 = Z;
k_delete = w * Np/Nc;
for i = 1:Nc
    for j = 1:Nq
        if (T(i,1)<= k_delete) | (C(i,1)<= Avms & max(S(i,:))<= Fstd)
                                                     % 删除条件
            Y3(i,1).feature(1,:) = [zeros(1,Nq)];    % 删除的聚类中心特征值赋 0
            N_delete = N_delete + 1;                 % 删除次数 + 1
            Nc_delete = Nc_delete - 1;               % 删除后的聚类中心数 - 1
            if (Nc_delete <= 2) | (Nc_delete >= 8)   % 分类数不满足要求时,跳出循环
                flag3 = 1;                           % 中间标志为 1,跳出循环
                break;
            end
        end
    end
    if flag3 == 1;                                   % 中间标志为 1,跳出循环
        break;
    end
end
%类别调整后的聚类中心特征值
Y4 = Z;
for i = 1:Nc
    if Y1(i,1).feature(1,:)~ = Y4(i,1).feature(1,:)
        Z(i,1).feature(1,:) = Y1(i,1).feature(1,:);
    elseif Y2(i,1).feature(1,:)~ = Y4(i,1).feature(1,:)
        Z(i,1).feature(1,:) = Y2(i,1).feature(1,:);
    elseif Y3(i,1).feature(1,:)~ = Y4(i,1).feature(1,:)
        Z(i,1).feature(1,:) = Y3(i,1).feature(1,:);
    end
end
if Nc_divide > Nc
    for i = Nc + 1:Nc_divide
        Z(i,1).feature(1,:) = Y2(i,1).feature(1,:);
    end
```

```
    end
    Y5 = Z;
    N1 = 0;                                  % 已删除的聚类中心个数
    for i = 1:Nc_divide
        if Y5(i,1).feature(1,:) == [zeros(1,Nq)]
            for i1 = i - N1:Nc_divide - N1    % 删除特征值为 0 的聚类中心
                Z(i1,1).feature(1,:) = Y5(i1 + N1,1).feature(1,:);
            end
            N1 = N1 + 1;
        end
    end
    % 类别调整后的分类数
    Nc = Nc - N_combine + N_divide - N_delete;
```

程序运行完之后，出现如图 5-32 所示的 59 组数据分类图。

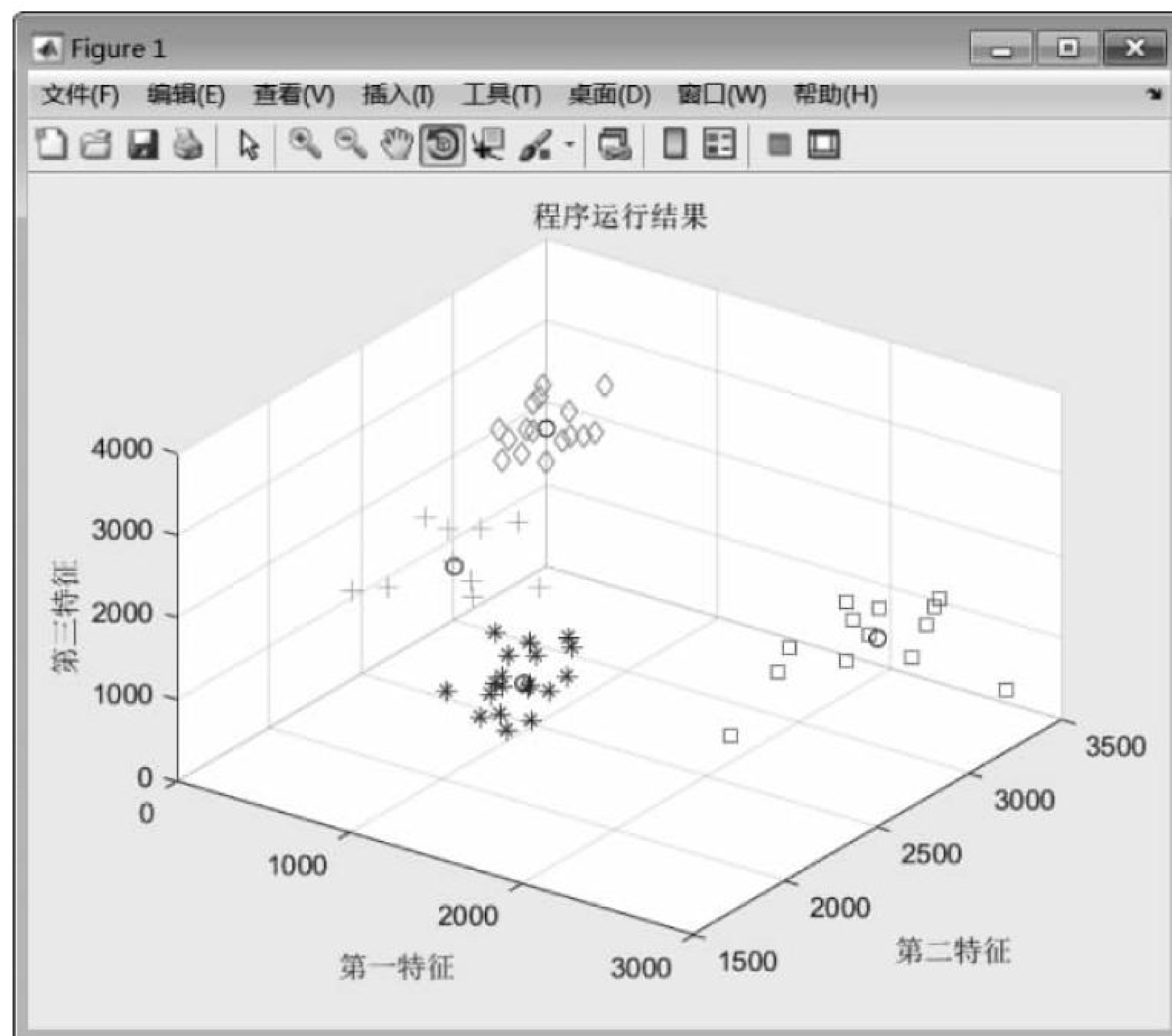


图 5-32　测试样本分类图

MATLAB 程序的运行结果如下：

```
A =
    1.0e+03 *
    1.7443    1.7519    1.9495
    0.3120    3.2136    2.2506
    2.2940    3.1569    1.0036
    1.2553    1.8340    2.9638
B' =
1 ～ 27 列
```

```
1    2    1    4    2    4    1    3    2    2    1    2    2    3
3    4    1    3    3    1    1    3    2    3    4    1    2
28 ～ 54 列
2    2    1    1    4    1    2    3    3    1    2    4    1    1
4    3    2    3    2    1    2    3    3    1    1    4    4
55 ～ 59 列
2    4    1    1    1
```

其中 A 为聚类中心，B'为分类结果。

5.8.5 结论

模糊 ISODATA 聚类分析方法对特性比较复杂而人们又缺少认识的对象进行分类，可以有效地实施人工干预，加入人脑思维信息，使分类结果更符合客观实际，给出相对的最优分类结果，因而具有一定的实用性。

然而由于该方法在计算中需要人为选择和确定不同的参数，使得该方法在数学理论上显得不够严谨。参数的选取也缺乏理论依据，选取最合适的参数也非常困难。这些参数的设定问题，直接影响到模糊分类的分类精度和算法实现，使得模糊 ISODATA 算法在实际应用中受到限制。

5.9 模糊神经网络

随着模糊信息处理技术和神经网络技术研究的不断深入，将模糊技术与神经网络技术进行了有机融合，构造出一种可“自动”处理模糊信息的神经网络——模糊神经网络。

5.9.1 模糊神经网络的应用背景

从模糊信息处理的角度来讲，自从模糊集合理论提出至今，有关模糊信息处理的理论和应用研究已取得了重大的发展，各种基于模糊逻辑和模糊信息处理技术的智能产品，已经走进各个工业控制领域及人们的消费生活中。但是作为模糊信息处理的核心“模糊规则的自动处理”及“模糊变量基本状态隶属度函数的自动生成”问题，却一直是困扰模糊信息处理技术进一步推广的两大难题。在过去，这些工作主要靠开发者的智慧和经验来进行。人们根据自己的经验，建立一套实用的规则及隶属函数，并从实践中检验，看与实际系统的性能要求是否符合，如果不符，则通过试探的方法对规则和隶属函数进行调整，直到满足性能要求为止。但是正确的调整并不是一件容易的事，这一工作往往需要很长时间和反复探索才能完成。

然而，以非线性大规模并行处理为主要特征的神经网络技术的出现，凭借其强大的自学习功能，帮助模糊推理系统解决了“模糊规则的自动处理”及“模糊变量基本状态隶属度函数的自动生成”问题。本节将采用模糊神经网络离线的从学习样本数据中自动提取参数优化

后的模糊参考模型,实现模糊推理系统合理、正确的建模。

5.9.2 模糊神经网络算法的原理

模糊推理系统类型的基本结构是一个模型,它将输入特性映射为输入隶属函数、输入隶属函数映射为规则、规则映射为一组输出特性、输出特性映射为输出隶属函数、输出隶属函数映射为一个单值输出或与输出相关的决策。因此,输入输出变量空间的划分、变量模糊集隶属函数的确定以及规则的个数、形式和各模糊算子 AND/OR 等的定义,对于模糊推理系统的建模至关重要。

假设要将模糊推理应用于一个系统,对该系统我们已经收集了用于建模、模型跟随或模式识别的输入输出数据。但是,在某些情况下,不能根据数据就辨识出隶属函数的形状及决定语言变量的个数。即使对给定的隶属度函数也不能任意选择参数,参数的选择应使隶属度函数适应输入/输出数据。这时,就需借助模糊逻辑工具箱中的 ANFIS 的被称为神经自适应学习技术选取最优的参数。

1. ANFIS 编辑器简介

神经自适应学习技术为模糊建模过程学习一个数据集的信息提供了一种方法,为计算隶属度函数参数最好允许相关的模糊推理系统跟踪给定的输入/输出数据(及自动调节隶属度函数的参数)。完成这一隶属度函数参数调节的模糊逻辑工具箱的函数是 anfis。可以从命令行或通过 ANFIS 编辑器 GUI 调用 anfis。为了操作简捷,这里选择 ANFIS 编辑器 GUI 来实现 ANFIS 推理。

自适应神经模糊推理系统(adaptive neuro-fuzzy inference system,ANFIS),通过一给定的输入/输出数据集,利用 ANFIS 编辑器 GUI 构建一个模糊推理系统(FIS),与隶属度函数相关的参数将通过学习过程来改变。ANFIS 或者单独使用反向传播算法或者结合最小二乘法一起进行隶属度函数参数的预测与优化。

ANFIS 比模糊推理系统复杂,并且并不是对所有的模糊推理系统都可用。特别地,ANFIS 只支持 Sugeno 型系统,并且还要满足:

(1) 一阶或零阶 Sugeno 型系统。

(2) 单输出,使用加权平均反模糊化法得到(线性或恒值输出隶属度函数)。

(3) 每条规则的权值为 1。

如果建模的 FIS 结构不遵守这些约束将产生错误。

进一步,ANFIS 不能接受基本模糊推理允许的所有定制选项,即不能生成自己的隶属度函数和反模糊化函数,只能使用它所提供的一种。

下面给出了一种用神经网络实现的基于 Takagi-Sugeno 型模糊系统的结构,将神经网络的学习功能引入到模糊推理系统中,通过自学习的过程来修正隶属度函数,以提高系统的自适应能力,达到满意的期望值。

2. Takagi-Sugeno 模型

Takagi-Sugeno 型模糊推理计算简单,易于数学分析。与其他类型的模糊推理方法不

同，Takagi-Sugeno 型模糊推理将去模糊化也结合到模糊推理中，其输出为精确量。一阶 Takagi-Sugeno 型模糊规则表达及计算公式如下

$$R^{i}: \mathrm{IF}\, x_1\, \mathrm{IS}\, A_{1j}\, \mathrm{AND}\, x_2\, \mathrm{IS}\, A_{2k}\, \mathrm{AND}\, x_3\, \mathrm{IS}\, A_{3k}\, \mathrm{THEN}\, y\, \mathrm{IS}\, f_i \tag{5-58}$$

$$\mu_i = A_{1j}(x_1)\cdot A_{2k}(x_2)\cdot A_{3k}(x_3) \tag{5-59}$$

$$\overline{\mu_i} = \frac{\mu_i}{\sum_{k=1}^{p}\mu_k} \tag{5-60}$$

$$y^* = \sum \overline{\mu_i} f_i \tag{5-61}$$

其中 A_{1j}, A_{2k}, A_{3l} 为模糊变量，$A_{1j}(x_1), A_{2k}(x_2), A_{3l}(x_3)$ 为隶属函数，f_i 为常数。

设 $j=1,2,\cdots,j_0$；$k=1,2,\cdots,k_0$；$l=1,2,\cdots,l_0$。j_0, k_0, l_0 是模糊子集个数，则 $p=j_0\cdot k_0\cdot l_0$ 为模糊规则最大条数。若取 $j_0=k_0=l_0=3$，则有 $p=27$。

3. 模糊神经网络的结构与学习算法

利用 ANFIS 构造的模糊神经网络结构如图 5-33 所示。该网络由前件网络和后件网络两部分组成，前件网络由前 4 层构成，用来匹配模糊规则的前件，后件网络简化为最后一层，用来产生模糊规则的后件。

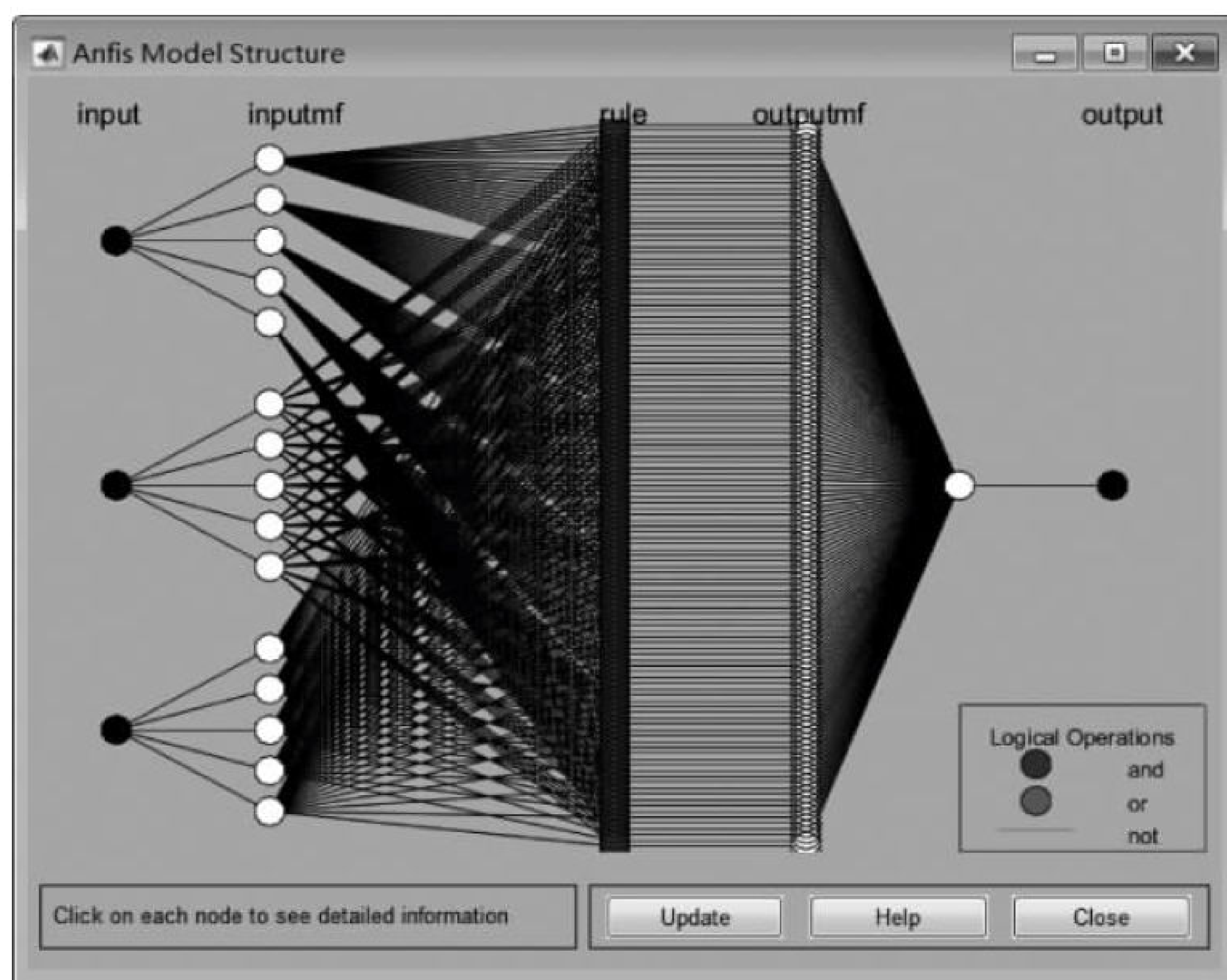


图 5-33 利用 ANFIS 构造的模糊神经网络结构

网络由 5 层构成，第 1 层为网络的模式输入层，输入节点是线性的，由 3 个神经元组成，将网络的输入信号 $x=[x_1,x_2,x_3]^{\mathrm{T}}$ 传送到下一层中。

第 2 层为网络的隐层，计算各输入分量属于语言变量值的模糊集的隶属函数 μ_i^j，其中 $\mu_i^j=\mu_{A_i^j}(x_i)(i=1,2,\cdots,n;\ j=1,2,\cdots,m_i)$。$n$ 是输入量维数 3，m_i 为输入量的模糊分割数 5；隶属函数为

$$\mu_i^j = \exp\left[\frac{-(x_i - c_{ij})^2}{\sigma_{ij}^2}\right] \tag{5-62}$$

式中,c_{ij} 和 σ_{ij} 分别表示隶属函数的中心和宽度。

第 3 层的每一个节点代表一条模糊规则,用来匹配模糊规则的前件,计算出每条规则的适用度。即

$$a_j = \min\{\mu_j^{i1}, \mu_j^{i2}, \mu_j^{i3}\}, \quad j = 1,2,\cdots,m;\ m = \prod_{i=1}^{n} m_i \tag{5-63}$$

$$i_1 \in \{1,2,\cdots,m_1\}, \quad i_2 \in \{1,2,\cdots,m_2\}, \quad i_3 \in \{1,2,\cdots,m_3\} \tag{5-64}$$

该层的节点总数 N3 $= m$,对于给定的输入,只有在输入点附近的语言变量值才有较大的隶属度值,远离输入点的语言变量值的隶属度或者很小或者为 0。当隶属度很小(例如小于 0.05)时近似取为 0。

第 4 层的节点数与第 3 层相同,即 N4=N3=m,它所实现的是归一化计算,即

$$\bar{\alpha} = \alpha_i / m \sum_{i=1} \alpha_i, \quad j = 1,2,\cdots,m \tag{5-65}$$

第 5 层是后件网络,用于计算每一条规则的后件,即

$$v_j = p_{j0} + p_{j1}x_1 + \cdots + p_{jn}x_n = \sum_{k=0}^{n} p_{jk}x_k, \quad j = 1,2,\cdots,m \tag{5-66}$$

每条规则的后件在简化结构中变成了最后一层的连接权,系统的输出为

$$v = \sum_{j=1}^{m} \bar{\alpha} v_j \tag{5-67}$$

可见 v 是各规则后件的加权和,加权系数为各模糊规则归一化的适用度,也即前件网络的输出用作后件网络的连接权值。

假设各输入分量的模糊分割数是预先确定的,那么需要学习的参数主要是后件网络的连接权 p_{ji}($j=1,2,\cdots,m$; $i=0,1,\cdots,n$)以及前件网络第二层各节点隶属函数的中心值 c_{ij}($j=1,2,\cdots,m$; $i=0,1,\cdots,m_i$)和宽度 σ_{ij}($j=1,2,\cdots,m$; $i=0,1,\cdots,m_i$)。

设误差代价函数为

$$E = \frac{1}{2}\sum_{i=1}^{r}(v_{di} - v_i)^2 \tag{5-68}$$

式中,v_{di} 和 v_i 分别表示期望输出和实际输出。

下面首先给出参数 p_{ji}^k 的学习算法

$$\frac{\partial E}{\partial p_{ji}^k} = \frac{\partial E}{\partial v_k}\frac{\partial v_k}{\partial v_{kj}}\frac{\partial v_{kj}}{\partial p_{ji}^k} = -(v_{dk} - v_k)\overline{\alpha_j}x_i \tag{5-69}$$

$$p_{ji}^k(l+1) = p_{ji}^k(l) - \beta\frac{\partial E}{\partial p_{ji}^k} = p_{ji}^k(l) + \beta(V_{dk} - v_k)\overline{\alpha_j}x_i \tag{5-70}$$

式中,$j=1,2,\cdots,m$; $i=0,1,\cdots,n$; $k=1,2,\cdots,r$; $\beta>0$ 为学习效率。

这时可将参数 p_{ji}^k 固定,利用误差反传算法来计算 $\frac{\partial E}{\partial c_{ij}}$ 和 $\frac{\partial E}{\partial \sigma_{ij}}$,再利用梯度寻优算法来调节 c_{ij} 和 σ_{ij},可得所求一阶梯度为

$$\frac{\partial E}{\partial c_{ij}} = \frac{\partial E}{\partial f_{ij}^{(2)}}\frac{\partial f_{ij}^{(2)}}{\partial c_{ij}} = -\delta_{ij}^{(2)}\frac{2(x_i - c_{ij})}{\sigma_{ij}{}^2} \tag{5-71}$$

$$\frac{\partial E}{\partial \sigma_{ij}} = \frac{\partial E}{\partial f_{ij}^{(2)}} \frac{\partial f_{ij}^{(2)}}{\partial \sigma_{ij}} = -\delta_{ij}^{(2)} \frac{2(x_i - c_{ij})}{\sigma_{ij}{}^3} \tag{5-72}$$

最后可给出参数调整的学习算法为

$$c_{ij}(l+1) = c_{ij}(l) - \beta \frac{\partial E}{\partial c_{ij}}, \quad i = 1,2,\cdots,m;\ j = 0,1,\cdots,m_i \tag{5-73}$$

$$\sigma_{ij}(l+1) = \sigma_{ij}(l) - \beta \frac{\partial E}{\partial \sigma_{ij}}, \quad i = 1,2,\cdots,m;\ j = 0,1,\cdots,m_i \tag{5-74}$$

其中，$\beta>0$ 为学习效率。

在设计中使用 BP 神经网络训练系统，其学习的主要参数为第 2 层各节点隶属函数的转折点，并依据输出误差优化隶属度函数。

5.9.3　模糊神经网络分类器的 MATLAB 实现

1. 样本数据的标准化

为提高运算速率及误差精度，需要对训练样本及测试样本进行数据的标准化。样本数据的标准化只对样本指标数据进行预处理，使其特征值映射到[0,1]区间上。设有 f 个样本 $x_1,x_2,\cdots,x_f$，每个样本 x_i 具有 n 个样本指标 $z_1,z_2,\cdots,z_n$；x_{ij} 表示第 i 个样本的第 j 个指标，f 个样本的 n 个指标可用表 5-6 表示。

表 5-6　样本指标数据

指　　标	z_1	z_2	z_3	…	z_n
x_1	x_{11}	x_{12}	x_{13}	…	x_{1n}
x_2	x_{21}	x_{22}	x_{23}	…	x_{2n}
…	…	…	…	…	…
x_f	x_{f1}	x_{f2}	x_{f3}	…	x_{fn}

f 个样本第 j 个指标的平均值 2 及标准差分别为

均值：
$$x_j = \frac{1}{f}\sum_{i=1}^{f} x_{ij} \tag{5-75}$$

标准差：
$$S_j = \sqrt{\frac{1}{f}\sum_{i=1}^{f}(x_{ij} - x_j)^2} \tag{5-76}$$

原始数据标准化为：
$$x'_{ij} = \frac{x_{ij} - x_j}{S_j} \tag{5-77}$$

运用极值标准化值公式，将标准化数据压缩到[0,1]内，即

$$x_{ij} = \frac{x'_{ij} - x'_{j\min}}{x'_{j\max} - x'_{j\min}} \tag{5-78}$$

式中，$x'_{j\min}$ 和 $x'_{j\max}$ 分别表示 $x'_{1j},x'_{2j},\cdots,x'_{fj}$ 中最小值和最大值；x_{ij} 为标准化后的指标，标准化后的样本如表 5-7 所示。

表 5-7　标准化后样本

样　本　号	特 征 向 量		
Record_#1	0.5866	0.0613	0.628
Record_#2	0.1194	0.7945	0.6393
Record_#3	0.5924	0.0493	0.3306
Record_#4	0.2874	0.0467	0.7192
Record_#5	0.068	0.7802	0.4952
Record_#6	0.292	0.2465	0.856
Record_#7	0.6083	0.0135	0.5494
Record_#8	0.7959	0.5193	0.2956
Record_#9	0.129	0.8842	0.5453
Record_#10	0.1106	0.9974	0.6505
Record_#11	0.529	0.0903	0.405
Record_#12	0.0277	0.9516	0.6367
Record_#13	0.1626	0.908	0.5605
Record_#14	0.7772	0.9258	0
Record_#15	0.7072	0.8412	0.0164
Record_#16	0.4768	0.1135	0.7552
Record_#17	0.6227	0.1877	0.5709
Record_#18	0.7457	0.8759	0.2252
Record_#19	1	0.8762	0.0428
Record_#20	0.5705	0.1611	0.5312
Record_#21	0.5663	0.0097	0.4015
Record_#22	0.95	0.7581	0.4893
Record_#23	0.0508	0.7931	0.6053
Record_#24	0.6972	0.853	0.2436
Record_#25	0.4873	0.0439	0.9687
Record_#26	0.5563	0.0811	0.3493
Record_#27	0.1086	0.7891	0.6425
Record_#28	0.0913	0.7988	0.5244
Record_#29	0.073	0.7896	0.5794
Record_#30	0.5738	0.043	0.5176
Record_#31	0.6338	0.1579	0.4859
Record_#32	0.2884	0.4038	0.675
Record_#33	0.6179	0.0811	0.361
Record_#34	0.1493	0.8919	0.5527
Record_#35	0.8037	0.9295	0.1485
Record_#36	0.7686	1	0.1387
Record_#37	0.6016	0.0212	0.5826
Record_#38	0.0598	0.8793	0.6446
Record_#39	0.5027	0.2677	0.6802
Record_#40	0.5378	0.1892	0.5371
Record_#41	0.5384	0.1889	0.3673
Record_#42	0.4168	0.1336	0.9808

续表

样　本　号	特 征 向 量		
Record_#43	0.7905	0.5623	0.3593
Record_#44	0.1128	0.9051	0.6205
Record_#45	0.7249	0.4904	0.0189
Record_#46	0.1375	0.804	0.5139
Record_#47	0.5071	0	0.4789
Record_#48	0.109	0.8902	0.5069
Record_#49	0.7447	0.8512	0.1351
Record_#50	0.755	0.7896	0.2577
Record_#51	0.532	0.1012	0.6507
Record_#52	0.6627	0.0198	0.4389
Record_#53	0.5029	0.2078	1
Record_#54	0.3765	0.0195	0.8109
Record_#55	0	0.9817	0.5432
Record_#56	0.4258	0.1837	0.7313
Record_#57	0.6079	0.0877	0.483
Record_#58	0.6131	0.1923	0.6053
Record_#59	0.6279	0.1172	0.4522

2. 建立BP神经网络

BP神经网络的MATLAB程序代码如下：

```
function f = bpfun()
 % Neural Network—bpfun.m
 % 输入矩阵的范围(数据源)
P = [20 3000;1400 3500;500 3500;];
 % 创建网络
net = newff(P,[6 1],{'tansig' 'purelin'});
 % 初始化神经网络
net = init(net);
 % 两次显示之间的训练步数,默认为25
net.trainParam.show = 50;
 % lr不能选择太大,太大了会造成算法不收敛,太小了会使训练时间太长
 % 一般选择在0.1～0.01
 % 训练速度
net.trainParam.lr = 0.05;
 % 训练次数,默认为100
net.trainParam.epochs = 2000;
 % 训练时间,默认为inf,表示训练时间不限
net.trainParam.time = 5000;
 % 训练的目标,默认为0
net.trainParam.goal = 0.001;
```

```
%建立源数据的矩阵
SourceDataConvert = importdata('bp_train_sample_data.dat');
SourceData = SourceDataConvert'
TargetConvert = importdata('bp_train_target_data.dat');
Target = TargetConvert'
%神经网络训练
net = train(net,SourceData,Target)
%显示训练后的各层权重
mat1 = cell2mat(net.IW(1,1))
mat2 = cell2mat(net.LW(2,1))
%mat3 = cell2mat(net.LW(3,2))
%读取仿真文件数据
simulate_data_convert = importdata('bp_simulate_data.dat');
simulate_data = simulate_data_convert';
result = sim(net,simulate_data)
result = result'
grid;
Title('BP神经网络：训练次数 - 输出误差对应关系图');
XLabel('训练次数');
YLabel('误差值');
%打开存储仿真结果的文件,a表示追加结果,w+表示清除原有内容后读或写
fid = fopen('result.dat','w+');
if fid == -1
    disp('file open error')
    fclosed(fid);
    return;
else
      ;
end
%仿真的样本数
simulate_num = 30;
%存储结果到文件中
for i = 1:1:simulate_num
    fprintf(fid,'%f',result(i));
    fprintf(fid,'%s','');
end
%换行输入
fprintf(fid,'%s\r\n','');
%fprintf(fid,'%f',20);
%关闭文件
fclose(fid);
%可以把结果文件的数据读出进行操作
matrix = importdata('result.dat');
```

由BP网络分出类的测试样本将被用于后面的检验。

3. 利用ANFIS编辑器GUI建模过程

在MATLAB命令行键入anfisedit,进入ANFIS编辑器的GUI图形编辑环境,如图5-34所示。

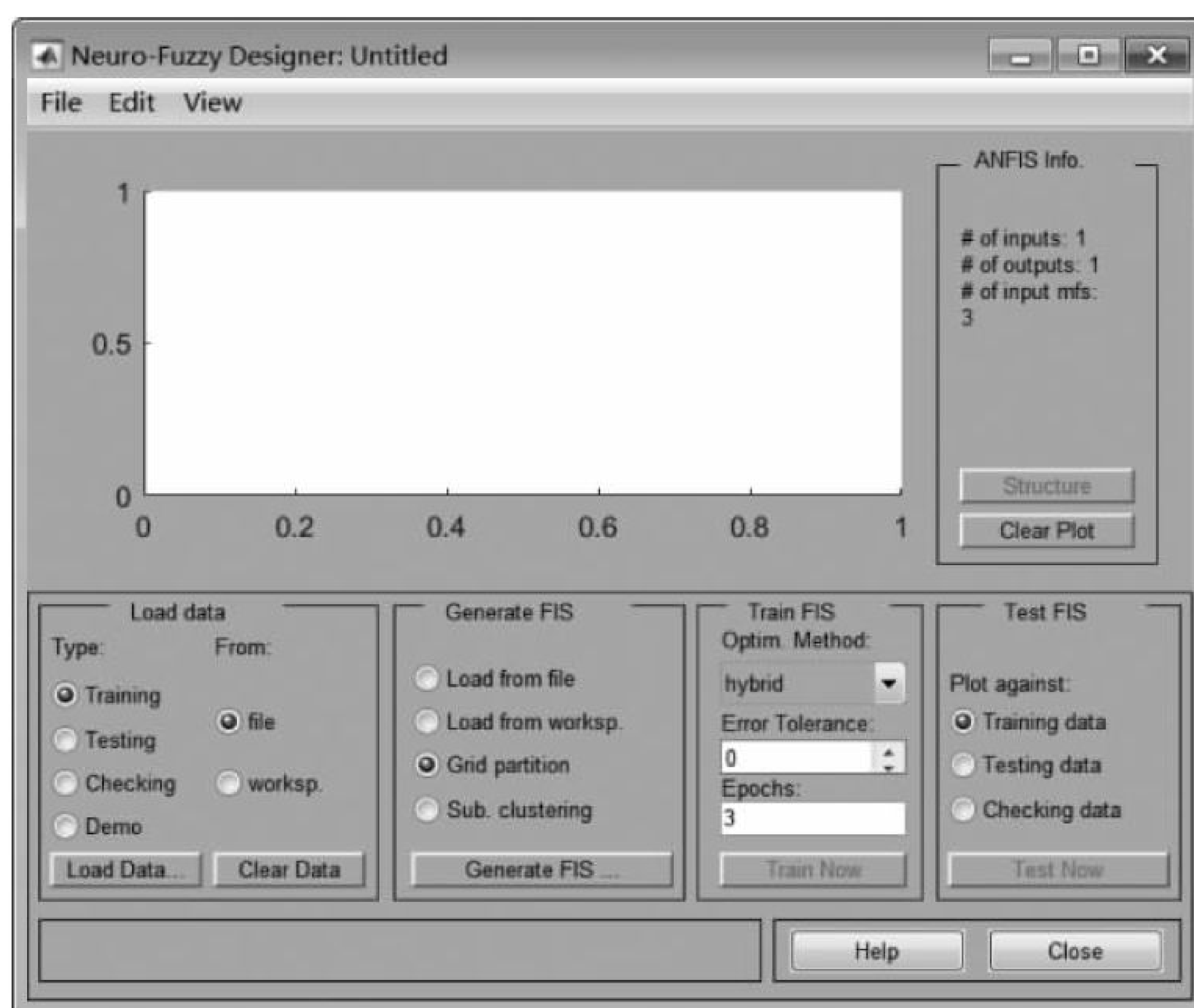


图 5-34　ANFIS 编辑器 GUI 图形界面

选择给定的已分类的数据作为训练样本，选择利用 BP 神经网络分出类的测试样本作为检验样本，然后选中单选按钮 Grid partition，单击 Generate FIS，即可出现一个基于 Sugeno 模型的 FIS 的界面，如图 5-35 所示。选择输入 Number of MFs 值为 5，设置 MF Type 为 gaussmf(高斯型)，选择输出为 constant，单击 OK 按钮即可生成一个 FIS，如图 5-36 所示。

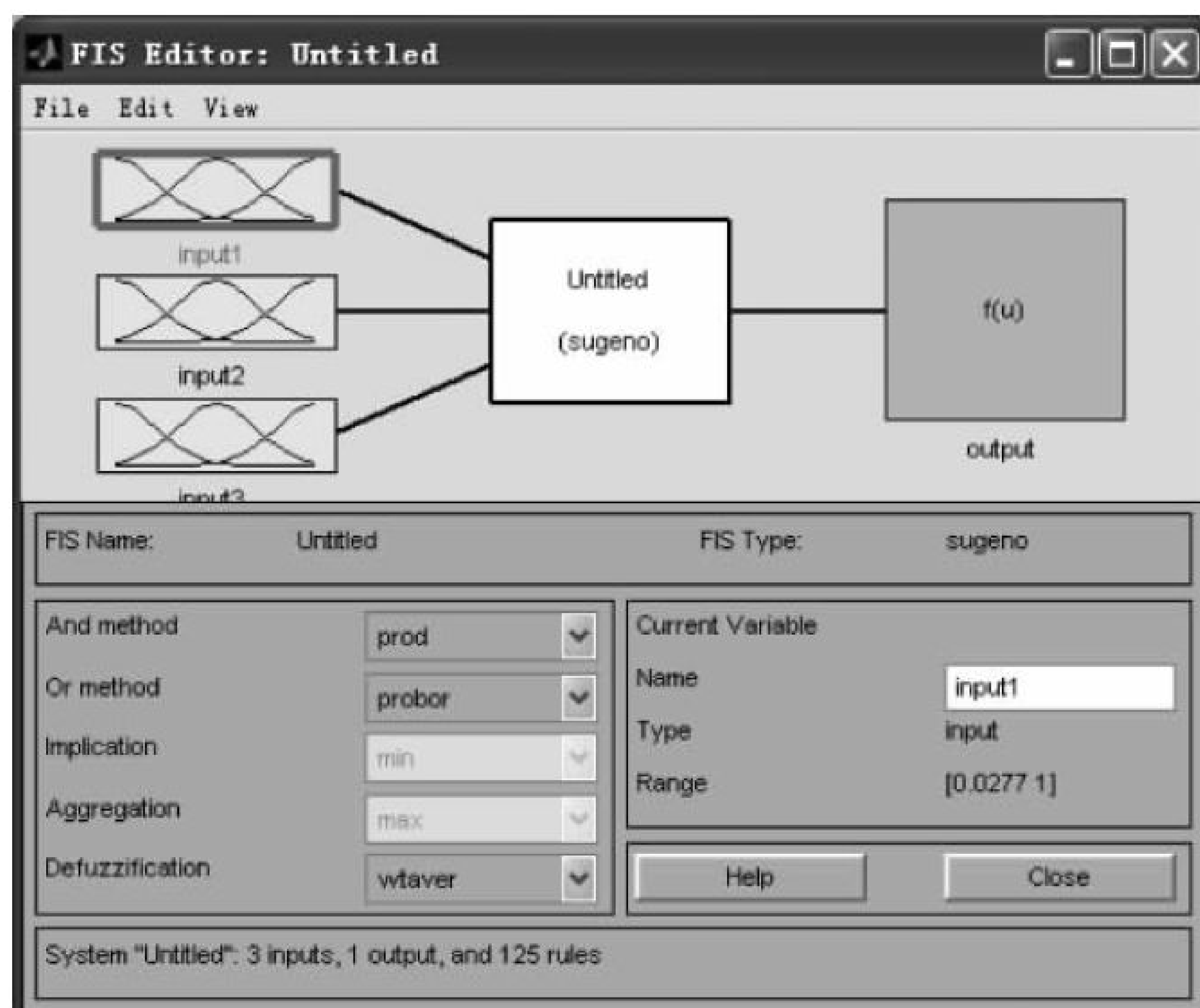


图 5-35　建立的 Sugeno 型的模糊推理系统



图 5-36　设置输入变量函数的个数、类型及输出类型

单击 Structure 可以观察模糊神经网络的结构,如图 5-37 所示。

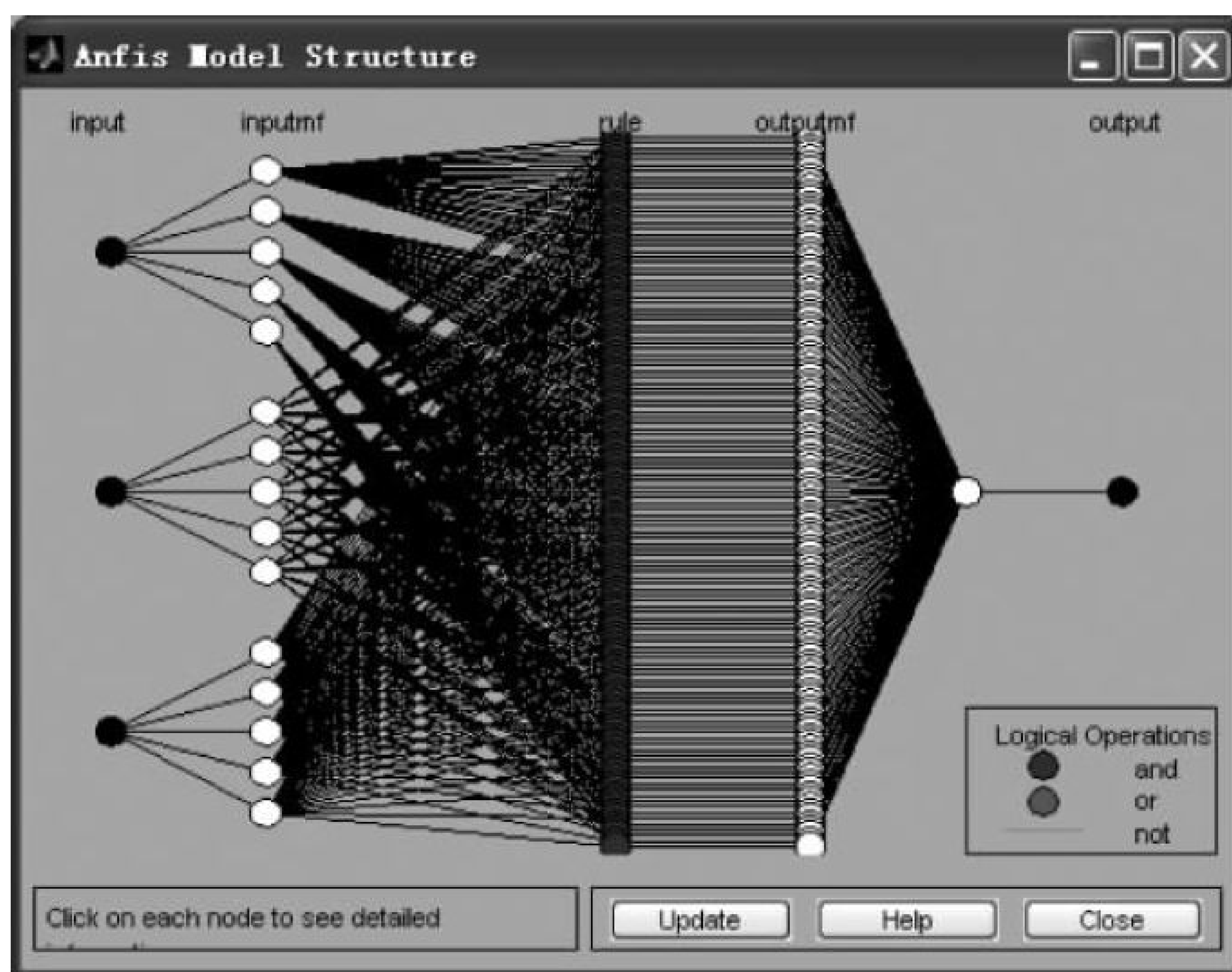


图 5-37　神经网络的结构图

选择 Optim. Method 为 backpropa, Error Tolerance 为 0.001, Epochs 为 1300 进行训练,即可通过神经网络对生成的 FIS 进行训练。逼近均方根误差曲线图如图 5-38 所示,当训练次数为 690 时,误差降到约为 0.0001。

训练样本经过 FIS 训练后,其输出散点图如图 5-39 所示。

使用经 BP 神经网络训练后分类的测试样本作为检测集,同样的测试样本经过 ANFIS 后其输出散点图与 BP 神经网络分类的散点对比图如图 5-40 所示。

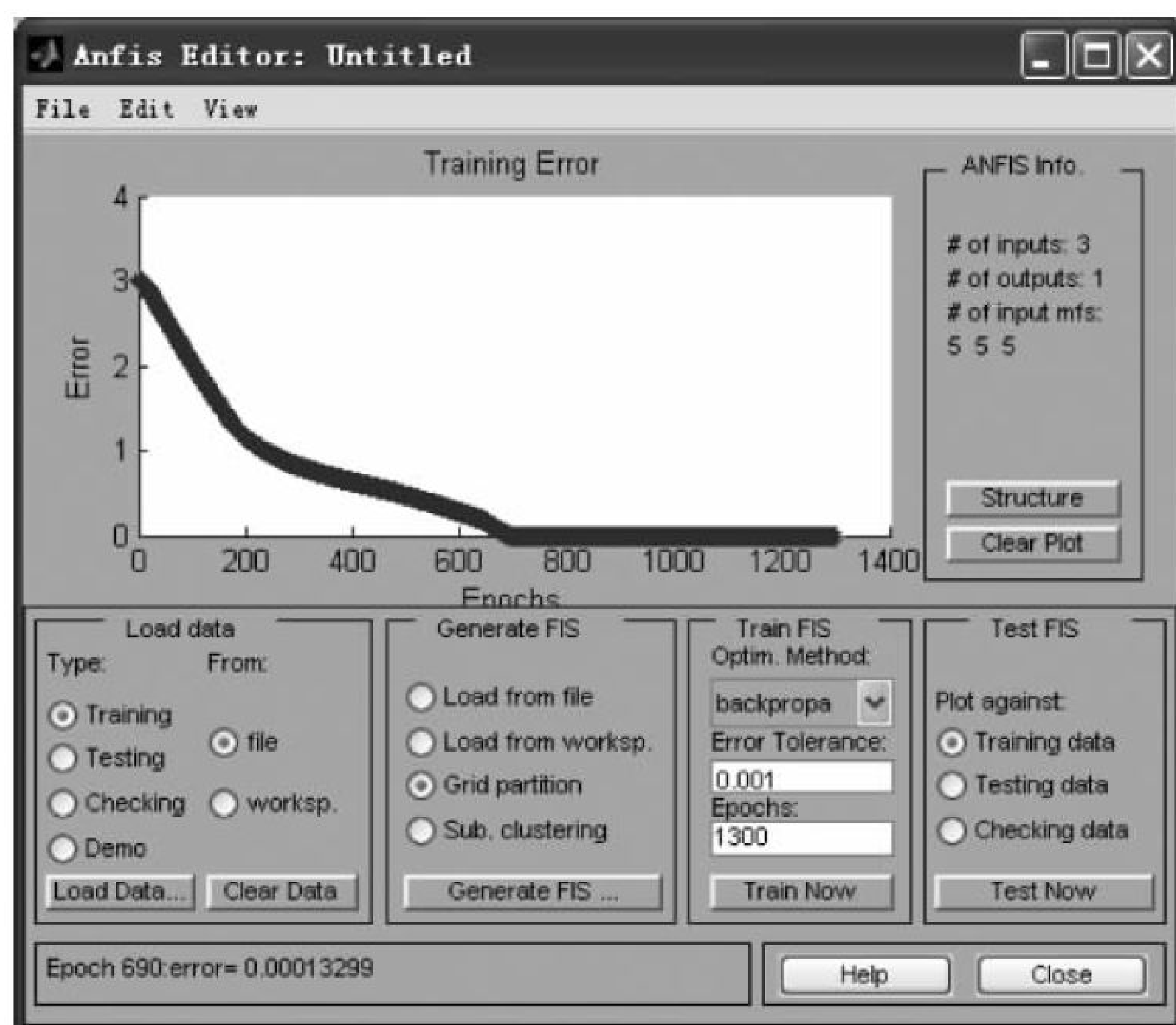


图 5-38 训练误差曲线图

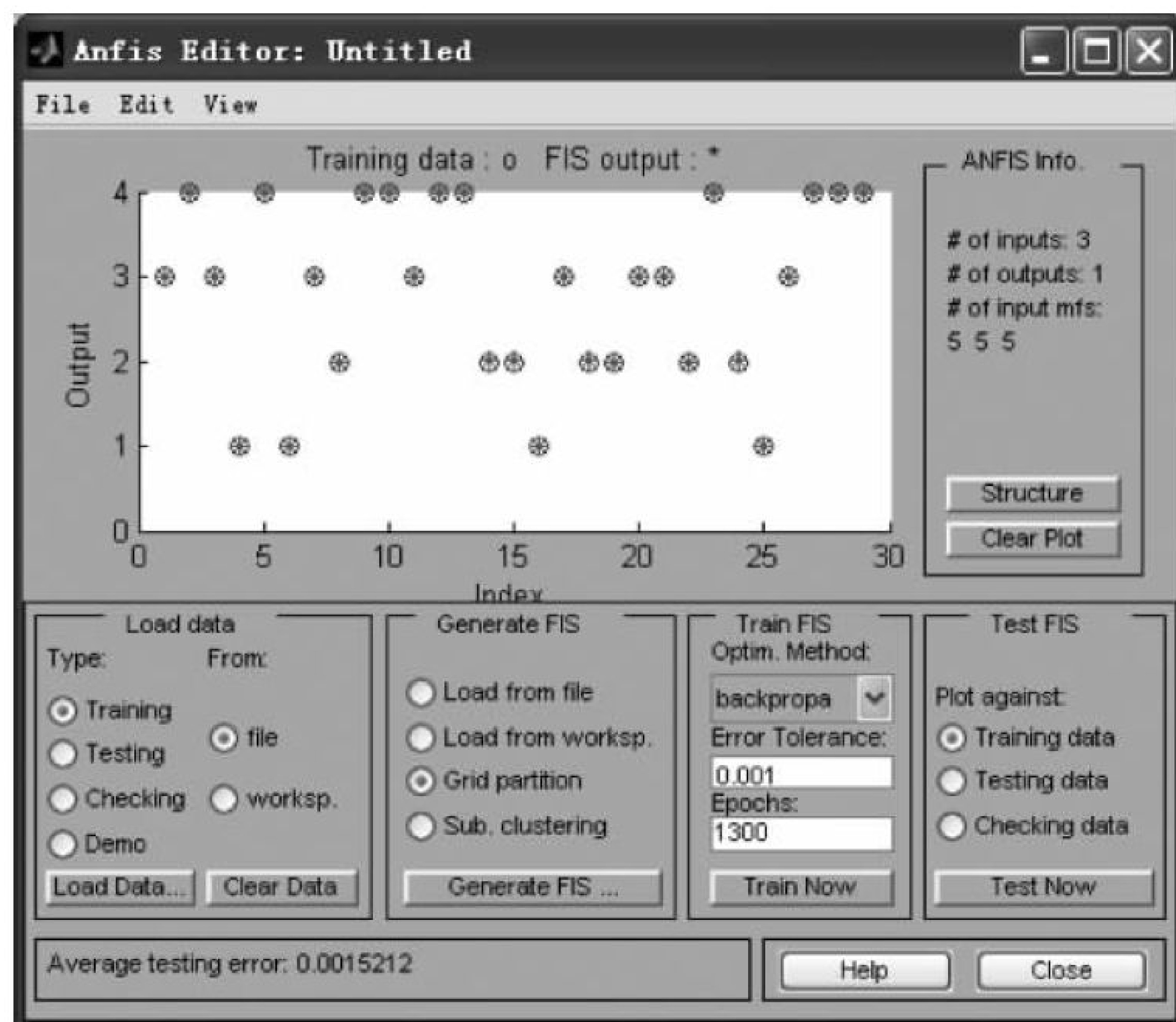


图 5-39 输出散点图

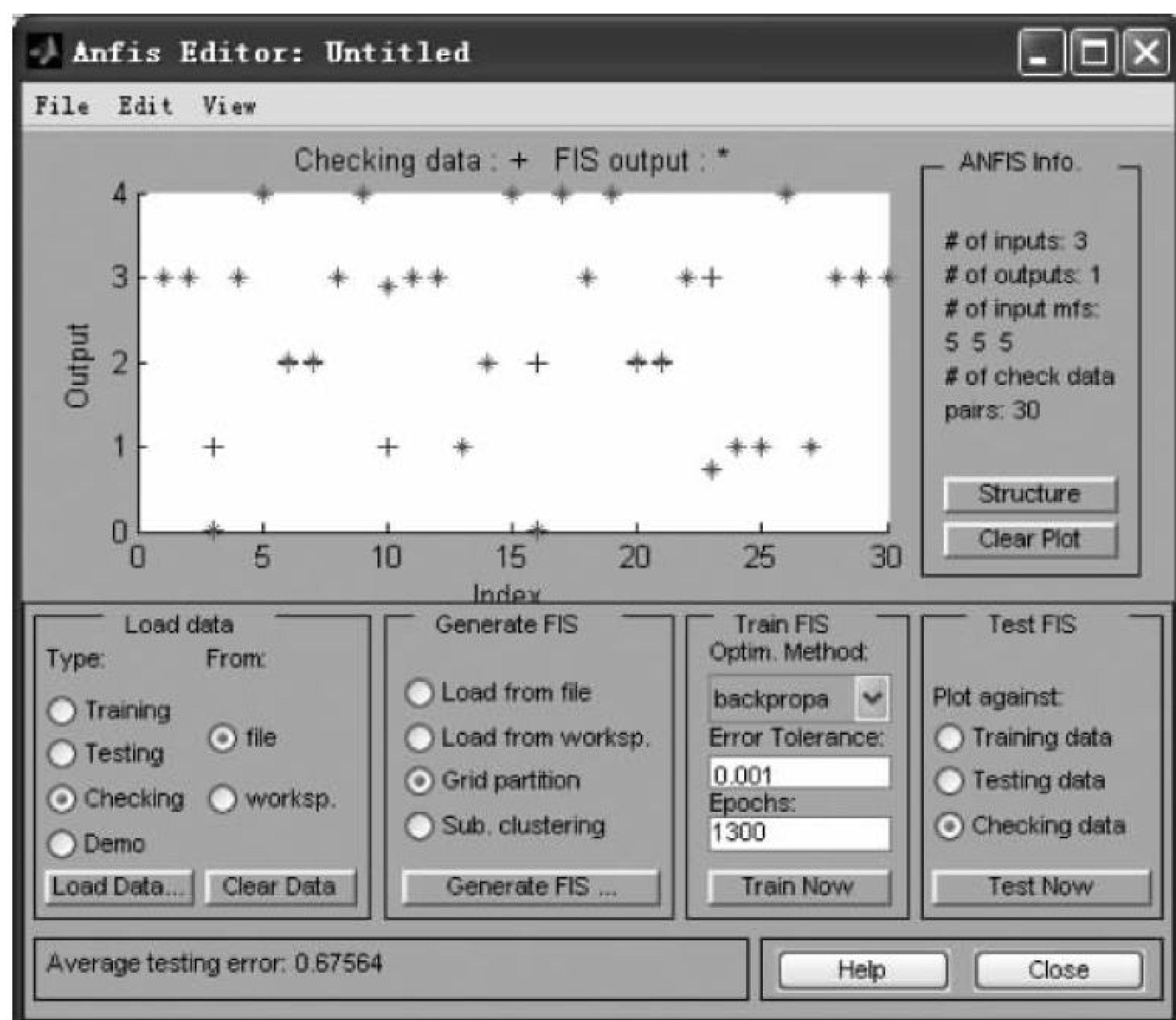


图 5-40 对比图

如图 5-40 所示,测试集中第 3、16、23 个样本的输出与 BP 神经网络分类不同,其他值均完全符合。经过反向误差传播算法优化后的 FIS 系统三个输入变量的隶属函数分别如图 5-41～图 5-43 所示,输出变量的隶属函数如图 5-44 所示。

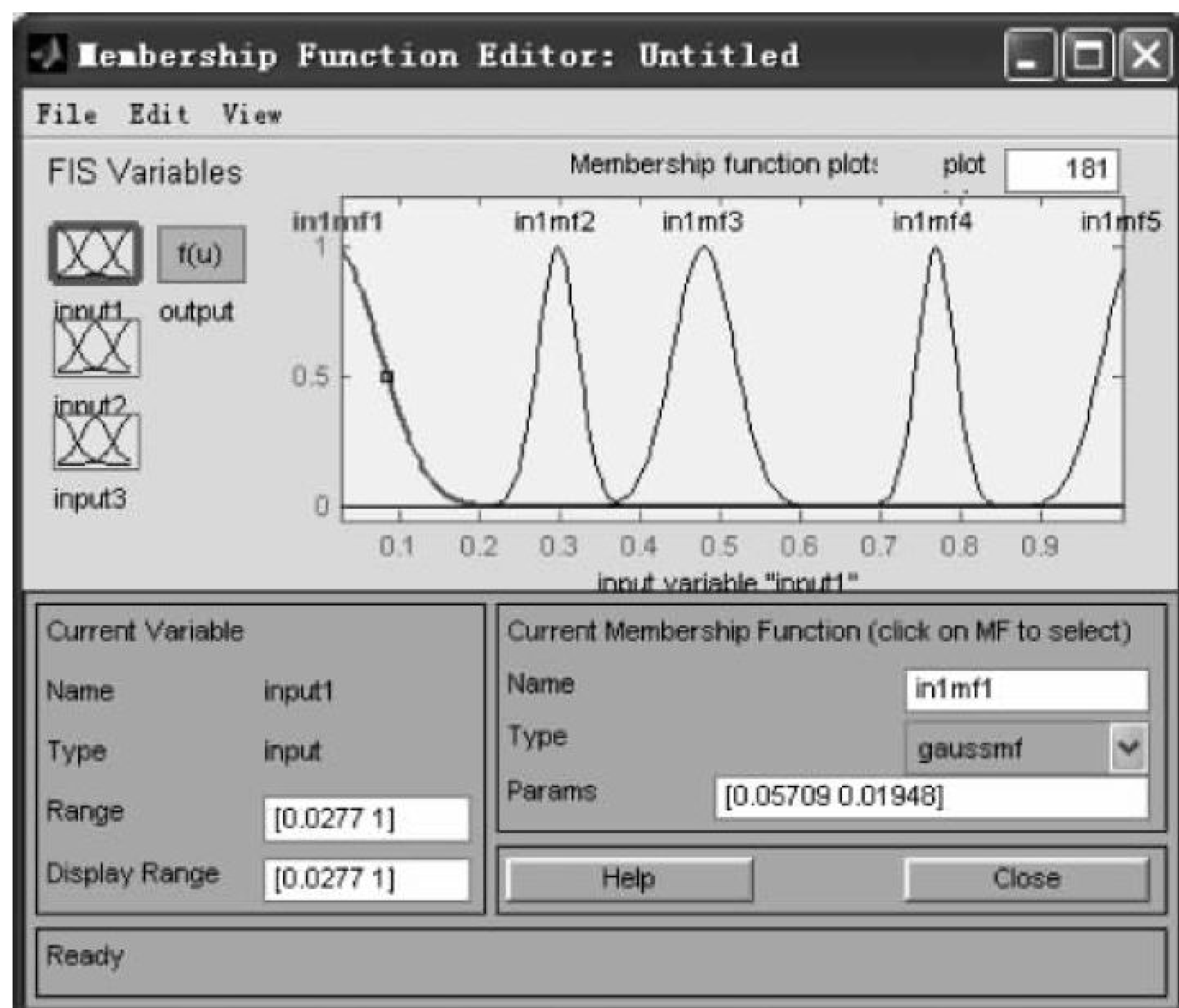


图 5-41 第 1 个输入变量优化后的隶属函数图形

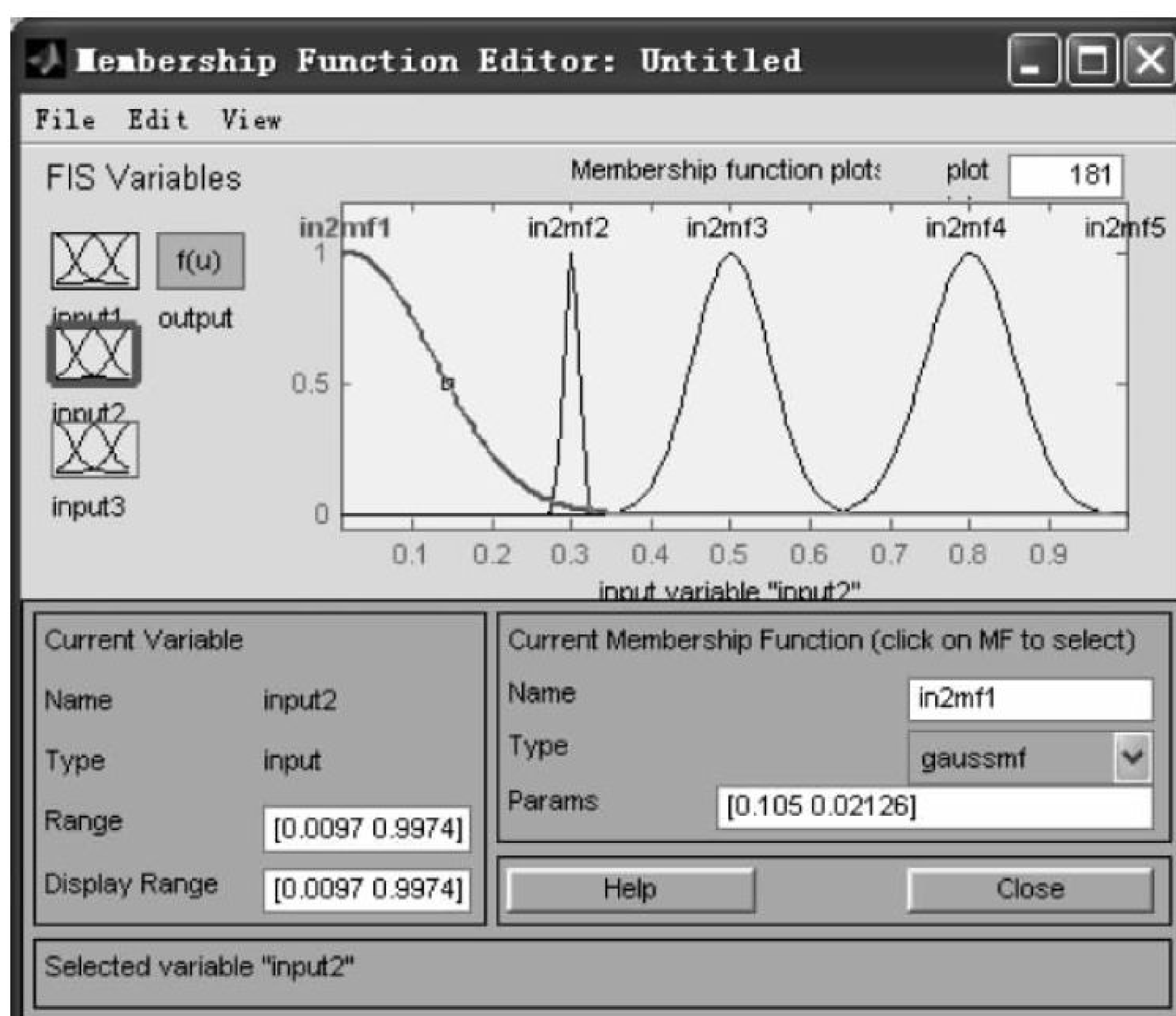


图 5-42 第 2 个输入变量优化后的隶属函数图形

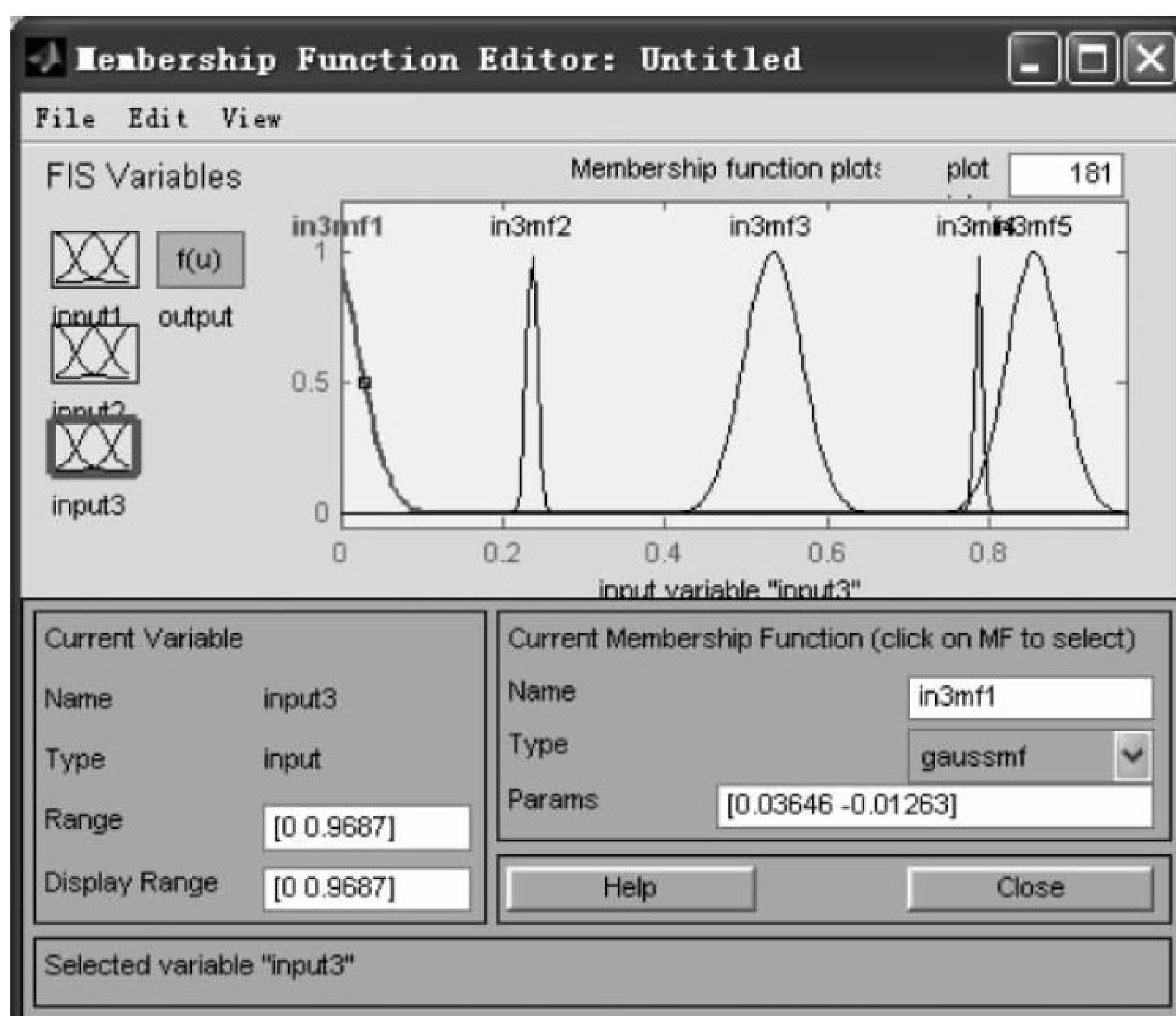


图 5-43 第 3 个输入变量优化后的隶属函数图形

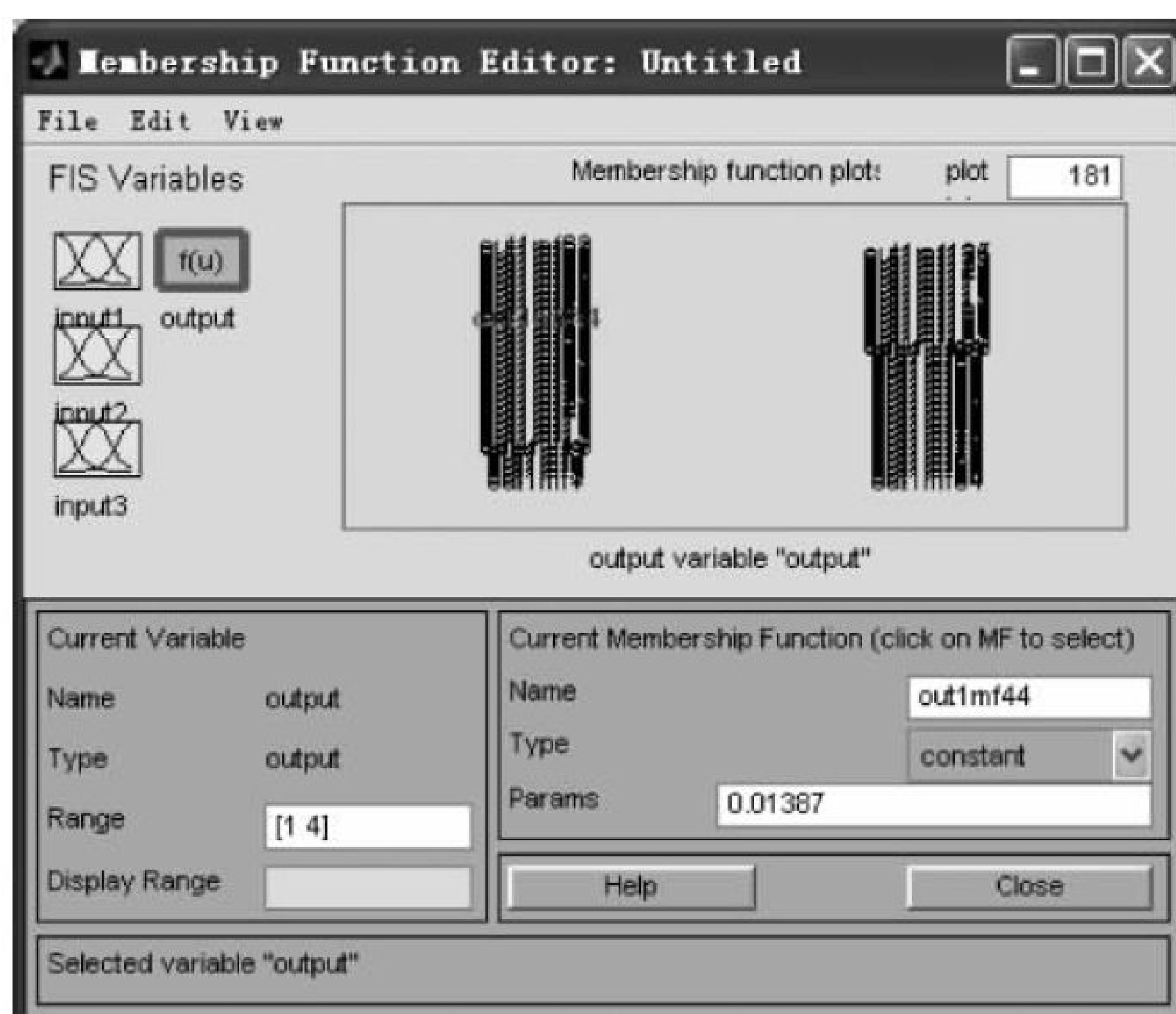


图 5-44 输出变量优化后的隶属函数图形

模糊规则如图 5-45 所示，其中所有规则的权重值均为 1。

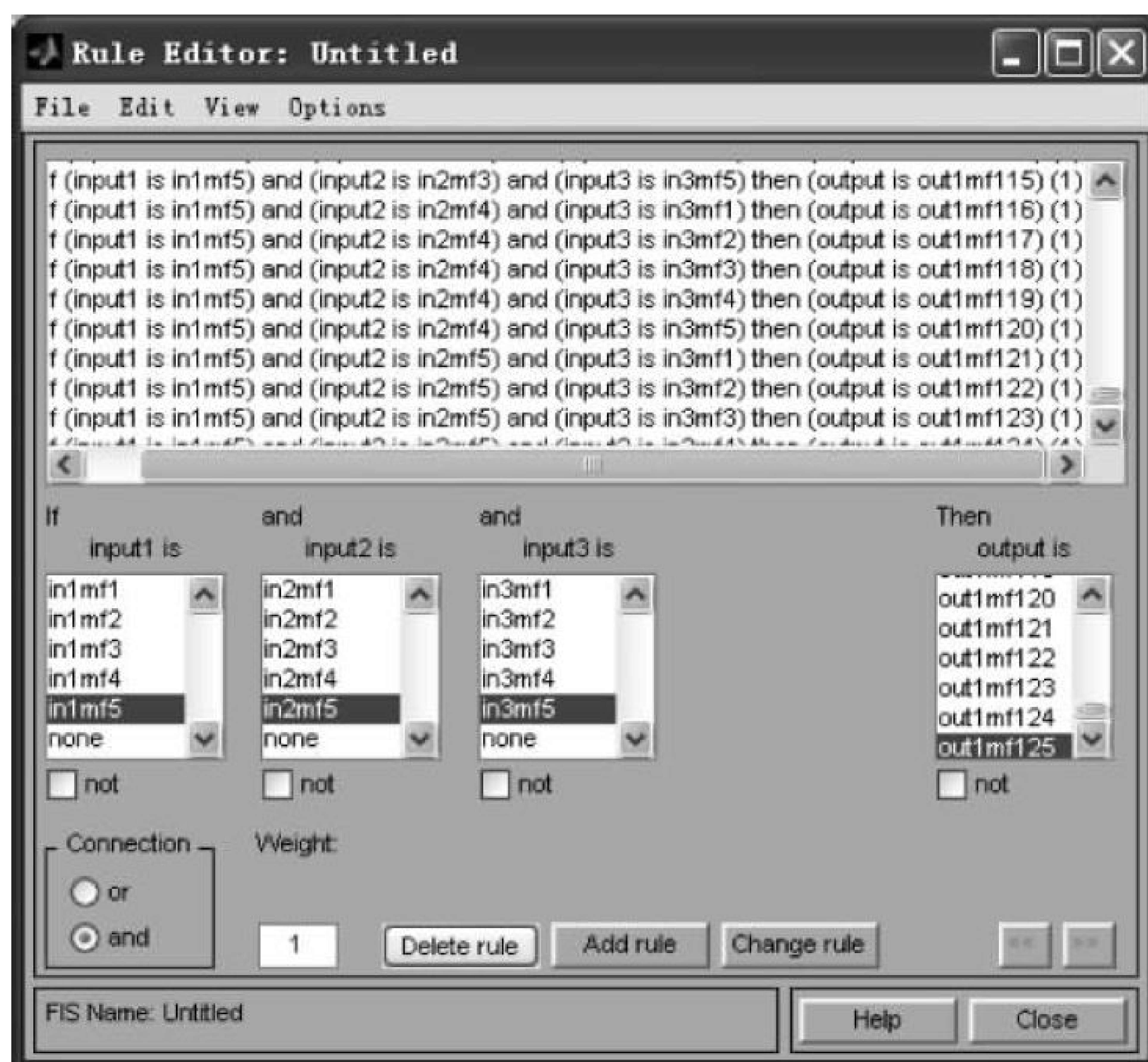


图 5-45 FIS 模糊规则图

规则编辑器如图 5-46 所示。

经过训练优化后的曲面观测图如图 5-47 所示。

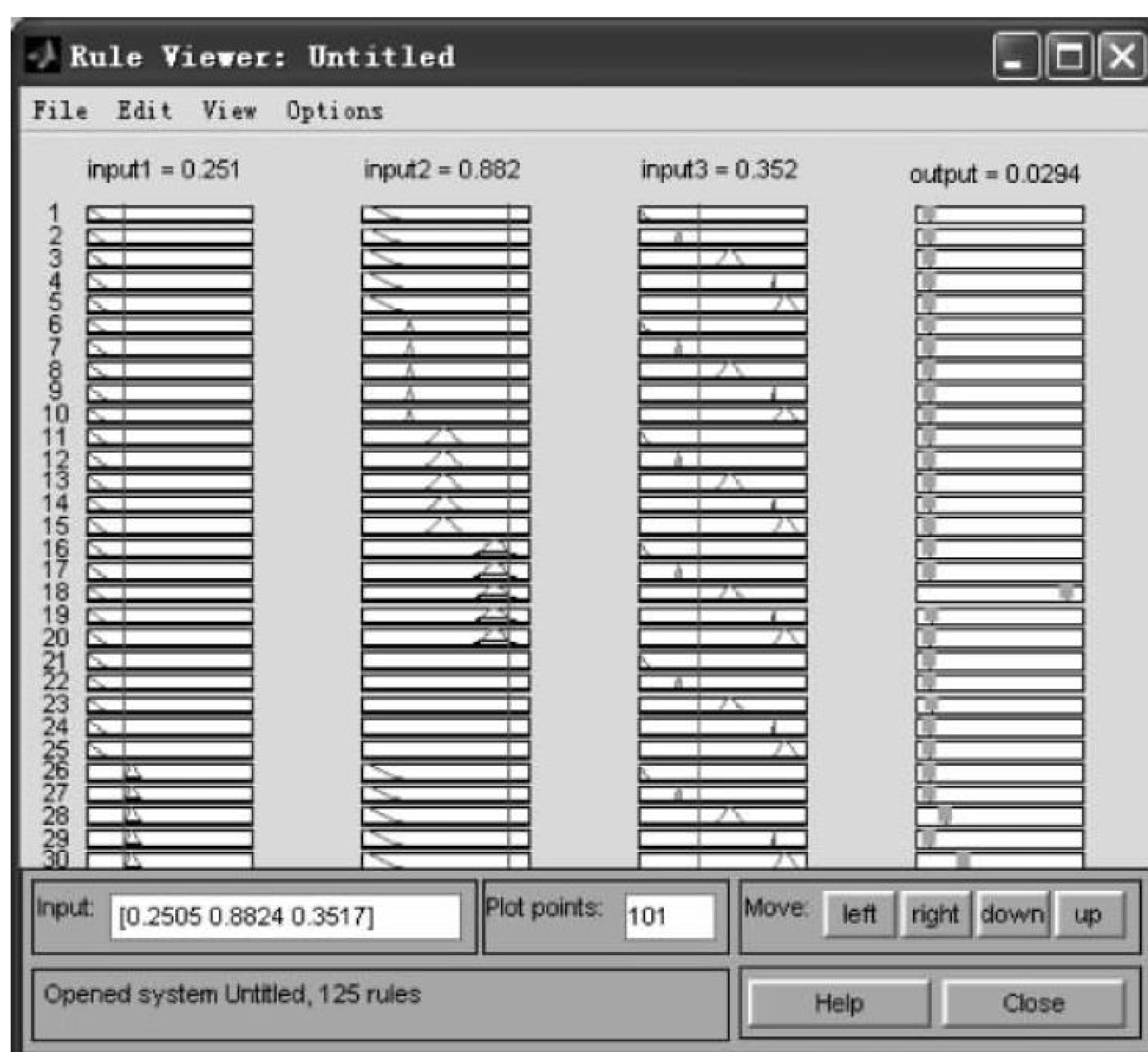


图 5-46 FIS 规则编辑器

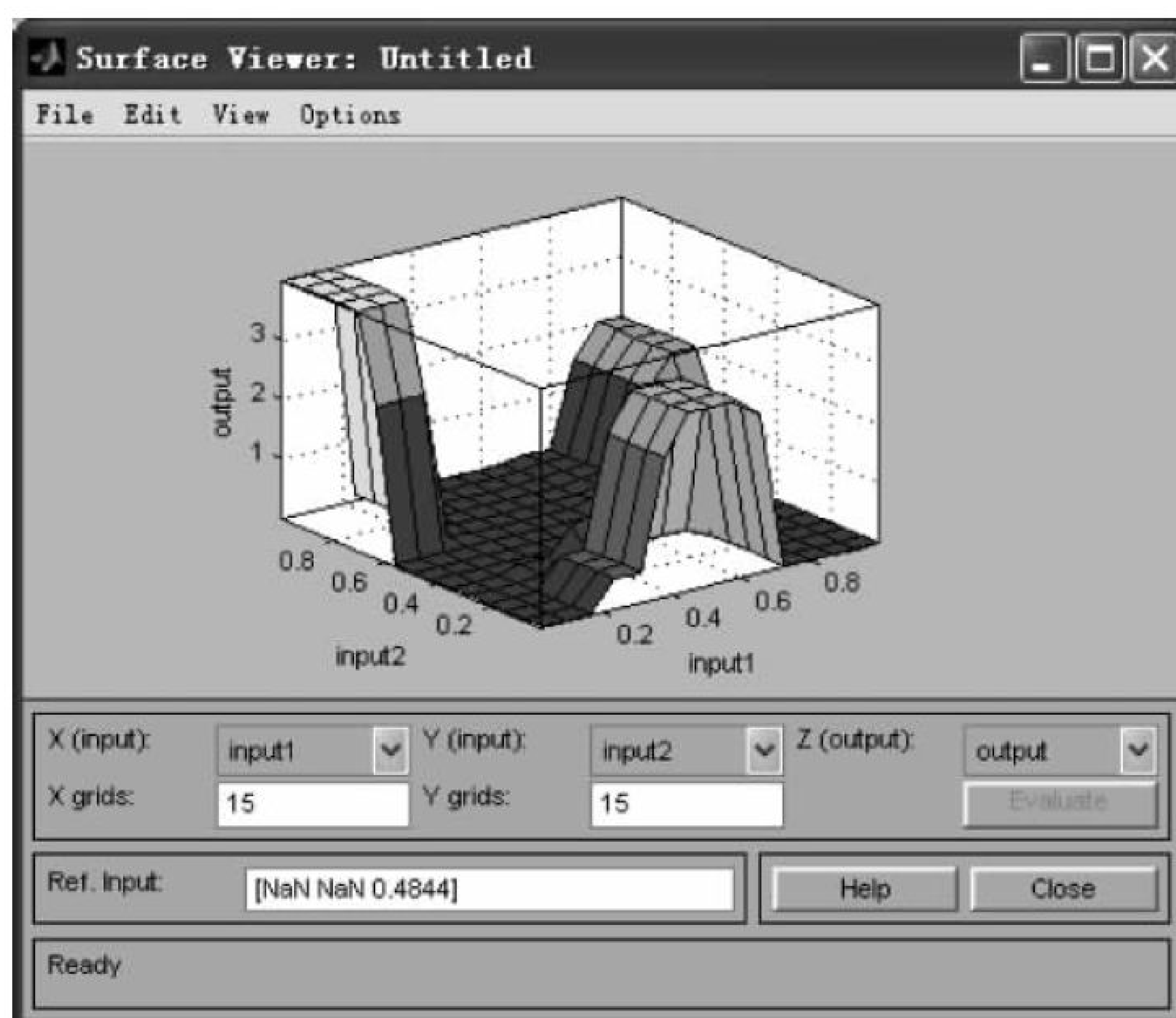


图 5-47 曲面观测图

4. 模糊神经网络分类结果

模糊神经网络的分类结果如表 5-8 所示。

表 5-8 模糊神经网络分类结果

特征向量			BP 模糊仿真结果	样本分类
1702.8	1639.79	2068.74	3	3
1877.93	1860.96	1975.3	3	3
867.81	2334.68	2535.1	未分出	1
1831.49	1713.11	1604.68	3	3
460.69	3274.77	2172.99	4	4
2374.98	3346.98	975.31	2	2
2271.89	3482.97	946.7	2	2
1783.64	1597.99	2261.31	3	3
198.83	3250.45	2445.08	4	4
1494.63	2072.59	2550.51	2.8	3
1597.03	1921.52	2126.76	3	3
1598.93	1921.08	1623.33	3	3
1243.13	1814.07	3441.07	1	1
2336.31	2640.26	1599.63	2	2
354	3300.12	2373.61	4	4
2144.47	2501.62	591.51	未分出	2
426.31	3105.29	2057.8	4	4
1507.13	1556.89	1954.51	3	3
343.07	3271.72	2036.94	4	4
2201.94	3196.22	935.53	2	2
2232.43	3077.87	1298.87	2	2
1580.1	1752.07	2463.04	3	3
1962.4	1594.97	1835.95	未分出	3
1495.18	1957.44	3498.02	1	1
1125.17	1594.39	2937.73	1	1
24.22	3447.31	2145.01	4	4
1269.07	1910.72	2701.97	1	1
1802.07	1725.81	1966.35	3	3
1817.36	1927.4	2328.79	3	3
1860.45	1782.88	1875.13	3	3

5.9.4 结论

模糊神经网络模型具有局部逼近功能,同时兼顾模糊与神经网络两者的优点;它既能模拟人脑的结构以及对信息的记忆和处理功能,擅长从输入输出数据中学习有用的知识,也能模拟人的思维和语言中对模糊信息的表达和处理方式,擅长利用人的经验性知识。对模糊聚类而言,模糊规则的建立一般是根据经验得来,因此其应用受到限制。将神经网络与模糊聚类相结合,利用神经网络自主学习的优点来建立模糊规则库,一方面提高了规则的可靠性,另一方面也打破了根据经验建立规则的局限性,扩大了其应用范围。

习题

（1）从模糊逻辑的发展过程看，模糊逻辑具有哪些特点？

（2）模糊逻辑描述的不确定性包含哪些？请举例说明。

（3）如何理解模糊集合与经典集合的关系？隶属度函数的引入对模糊系统有何意义？

（4）简述模糊 C 均值算法的原理。

（5）简述模糊神经网络的原理。

第6章

神经网络聚类设计

在 20 世纪的很长时间里,科学家们期望使用计算机系统模拟人类思维。大约 50 年过去了,研究者建立了第一个神经网络的电子硬件模型。从此,大量科学团体致力于新的数学模型和训练算法的研究。

6.1 什么是神经网络

从生物学的角度说,“神经”就是“神经系统”的缩写。“神经系统”是机体内起主导作用的系统,包括中枢神经和周围神经两部分。中枢神经通过周围神经与其他各个器官、系统发生极其广泛的联系。

那么这种联系是如何发生的呢?这与神经系统的生物结构有关,神经系统由神经细胞(神经元)和神经胶质组成。在人体的神经系统里,神经元的神经纤维主要集中在周围神经系统,其中许多神经纤维集结成束,外面包着由结缔组成的膜,就成为一条神经。它可以把中枢神经系统的兴奋传递给各个器官,也可把各个器官的兴奋传递给中枢神经系统的组织,这样就实现了中枢神经与人体其他各个器官、系统的联系。

上面所叙述的内容只是纯粹的生物学理论,那么又和控制理论中的“神经网络”有何关联?让我们从“神经网络”技术的发展历程寻找答案。

6.1.1 神经网络的发展历程

神经网络研究的主要发展过程大致可分为四个阶段。

1. 第一阶段(20 世纪 50 年代中期之前)

西班牙解剖学家 Cajal 于 19 世纪末创立了神经元学说,该学说认为神经元的形状呈两极,其细胞体和树突从其他神经元接受冲动,而轴突则将信号向远离细胞体的方向传递。在他之后发明的各种染色技术和微电极技术不断提供了有关神经元的主要特征及其电学

性质。

1943 年，美国的心理学家 W. S. McCulloch 和数学家 W. A. Pitts 在论文《神经活动中所蕴含思想的逻辑活动》中，提出了一个非常简单的神经元模型，即 M-P 模型。该模型将神经元当作一个功能逻辑器件来对待，从而开创了神经网络模型的理论研究。1949 年，心理学家 D. O. Hebb 写了一本题为《行为的组织》的书，在这本书中他提出了神经元之间连接强度变化的规则，即后来所谓的 Hebb 学习法则。Hebb 写道："当神经细胞 A 的轴突足够靠近细胞 B 并能使之兴奋时，如果 A 重复或持续地激发 B，那么这两个细胞或其中一个细胞上必然有某种生长或代谢过程上的变化，这种变化使 A 激活 B 的效率有所增加。"简单地说，就是如果两个神经元都处于兴奋状态，那么它们之间的突触连接强度将会得到增强。

20 世纪 50 年代初，生理学家 Hodykin 和数学家 Huxley 在研究神经细胞膜等效电路时，将膜上离子的迁移变化分别等效为可变的 Na＋电阻和 K＋电阻，从而建立了著名的 Hodykin-Huxley 方程。

这些先驱者的工作激发了许多学者从事这一领域的研究，从而为神经计算的出现打下了基础。

2. 第二阶段（20 世纪 50 年代中期到 60 年代末）

1958 年，F. Rosenblatt 等人研制出了历史上第一个具有学习型神经网络特点的模式识别装置，即代号为 Mark I 的感知机（perceptron），这一重大事件是神经网络研究进入第二阶段的标志。对于最简单的没有中间层的感知机，Rosenblatt 证明了一种学习算法的收敛性，这种学习算法通过迭代地改变连接权来使网络执行预期的计算。

稍后，Rosenblatt 和 B. Widrow 等人创造出了一种不同类型的会学习的神经网络处理单元，即自适应线性元件 Adaline，并且还为 Adaline 找出了一种有力的学习规则，这个规则至今仍被广泛应用。Widrow 还建立了第一家神经计算机硬件公司，并在 20 世纪 60 年代中期实际生产商用神经计算机和神经计算机软件。

除 Rosenblatt 和 Widrow 外，在这个阶段还有许多人在神经计算的结构和实现思想方面做出了很大的贡献。例如，K. Steinbuch 研究了称为学习矩阵的一种二进制联想网络结构及其硬件实现。N. Nilsson 于 1965 年出版的《机器学习》一书对这一时期的活动做了总结。

3. 第三阶段（20 世纪 60 年代末到 80 年代初）

第三阶段开始的标志是 1969 年 M. Minsky 和 S. Papert 所著的《感知机》一书的出版。该书对单层神经网络进行了深入分析，并且从数学上证明了这种网络功能有限，甚至不能解决像"异或"这样的简单逻辑运算问题。同时，他们还发现有许多模式是不能用单层网络训练的，而多层网络是否可行还很值得怀疑。

由于 M. Minsky 在人工智能领域中的巨大威望，他在论著中作出的悲观结论给当时神经网络沿感知机方向的研究泼了一盆冷水。在《感知机》一书出版后，美国联邦基金有 15 年之久没有资助神经网络方面的研究工作，苏联也取消了几项有前途的研究计划。

但是，即使在这个低潮期里，仍有一些研究者继续从事神经网络的研究工作，如美国波

士顿大学的 S. Grossberg、芬兰赫尔辛基技术大学的 T. Kohoneng 和日本东京大学的甘利俊一等人。他们坚持不懈的工作为神经网络研究的复兴开辟了道路。

4. 第四阶段(20 世纪 80 年代初至今)

1982 年,美国加州理工学院的生物物理学家 J. J. Hopfield 采用全互连型神经网络模型,利用所定义的计算能量函数,成功地求解了计算复杂度为 NP 完全型的旅行商问题(travelling salesman problem,TSP)。这项突破性进展标志着神经网络方面的研究进入了第四阶段,也是蓬勃发展的阶段。Hopfield 模型提出后,许多研究者力图扩展该模型,使之更接近人脑的功能特性。1983 年,T. Sejnowski 和 G. Hinton 提出了"隐单元"的概念,并且研制出了 Boltzmann 机。日本的福岛邦房在 Rosenblatt 的感知机的基础上,增加隐含层单元,构造出了可以实现联想学习的"认知机"。Kohonen 应用 3000 个阈器件构造神经网络,实现了二维网络的联想式学习功能。1986 年,D. Rumelhart 和 J. McClelland 出版了具有轰动性的著作《并行分布处理——认知微结构的探索》,该书的问世宣告神经网络的研究进入了高潮。

1987 年,首届国际神经网络大会在圣地亚哥召开,国际神经网络联合会(INNS)成立。随后 INNS 创办了刊物 Journal Neural Networks,其他专业杂志如 Neural Computation、IEEE Transactions on Neural Networks、International Journal of Neural Systems 等也纷纷问世。世界上许多著名大学相继宣布成立神经计算研究所并制定有关教育计划,许多国家也陆续成立了神经网络学会,并召开了多种地区性、国际性会议,优秀论著、重大成果不断涌现。

在经过多年的准备与探索之后,神经网络的研究工作已进入了决定性的阶段。日本、美国及西欧各国均制定了有关的研究规划。

从神经网络的发展历程可以得到以下两个结论。

(1) 控制理论中的"神经网络"是对生物神经系统的模拟,希望通过对生物神经系统智能工作过程的"物理"模拟,实现一个"智能"的"物理"系统。

(2) 控制理论中的"神经网络"的发展是人们在对生物神经系统的组织结构和功能机制进行深入探索研究的基础上不断发展的。

在这里,引用神经网络前人的话说"具有神经网络的'物理'系统,在微观上或者说在结构上能与生物神经网络取得拓扑一致,而在宏观功能上能与人类智能行为恰当对应"。

6.1.2 生物神经系统的结构及冲动的传递过程

神经系统是机体内起主导作用的系统。内、外环境的各种信息由感受器接受后,通过周围神经传递到脑和脊髓的各级中枢进行整合,再经周围神经控制和调节机体各系统器官的活动,以维持机体与内、外界环境的相对平衡。

神经系统由神经细胞(神经元)和神经胶质所组成。

神经元是一种高度特化的细胞,是神经系统的基本结构和功能单位,它具有感受刺激和传导兴奋的功能。神经元由胞体和突起两部分构成。胞体的中央有细胞核,核的周围为细胞质,细胞质内除有一般细胞所具有的细胞器(如线粒体、内质网等)外,还含有特有的神经

元纤维及尼氏体。神经元的突起根据形状和机能又分为树突和轴突。树突较短但分支较多，它接受冲动，并将冲动传至细胞体，各类神经元树突的数目多少不等，形态各异。每个神经元只发出一条轴突，长短不一，胞体发生出的冲动沿轴突传出。神经元的结构如图 6-1 所示。

突触的结构如图 6-2 所示。

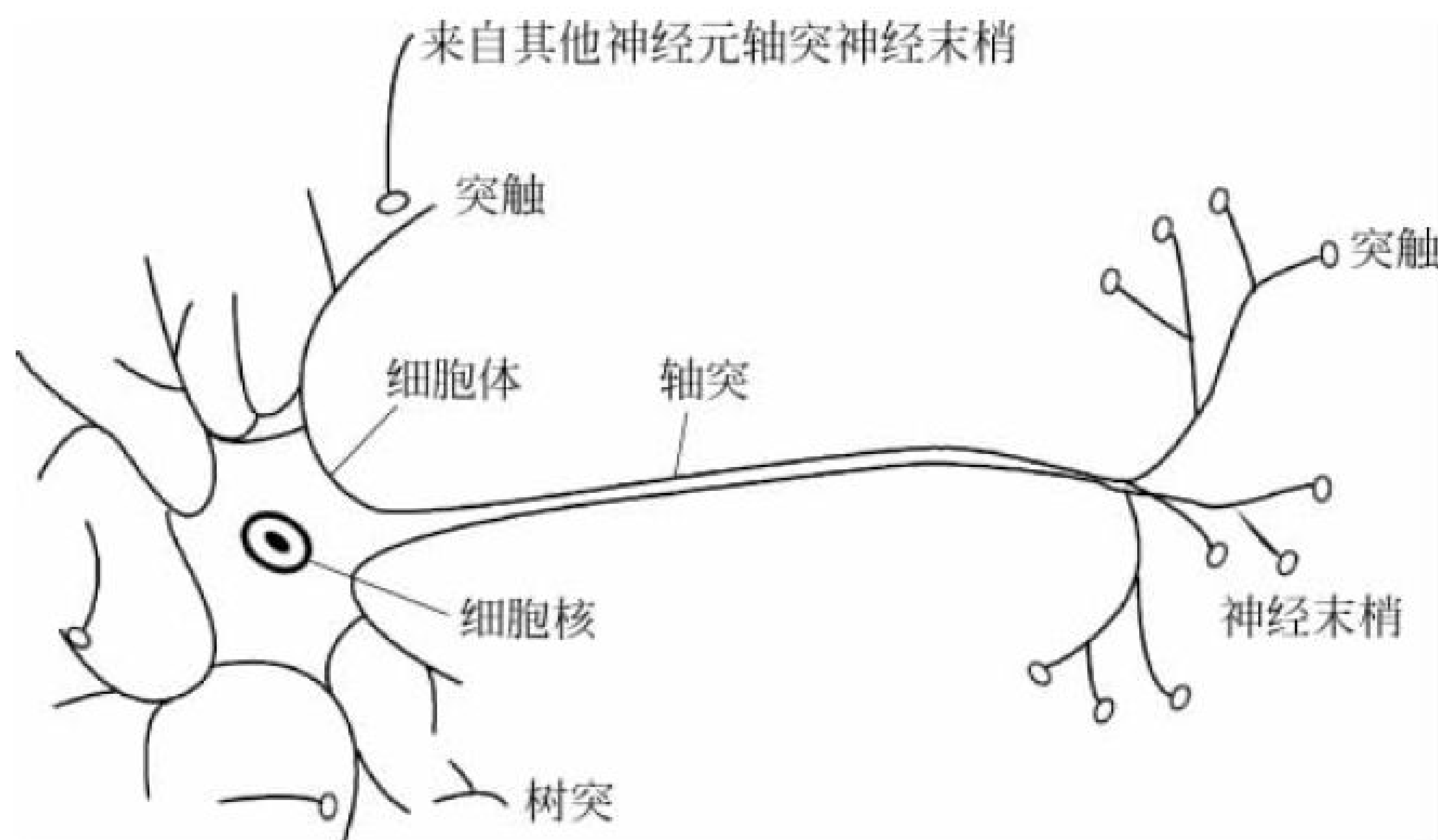


图 6-1 神经元的结构



图 6-2 突触的结构

突触传递冲动的过程如下：

(1) 神经冲动到达突触前神经元轴突末梢→突触前膜去极化。

(2) 电压门控Ca^{2+}通道开放→膜外Ca^{2+}内流入前膜。

(3) Ca^{2+}与胞浆 CaM 结合成 4 Ca^{2+}-CaM 复合物→激活 CaM 依赖的 PK Ⅱ→囊泡外表面突触蛋白Ⅰ磷酸化→蛋白Ⅰ与囊泡脱离→解除蛋白Ⅰ对囊泡与前膜融合及释放递质的阻碍作用。

(4) 囊泡通过出胞作用量子式释放递质入间隙(囊泡可再循环利用)。

(5) 神经递质→作用于后膜上特异性受体或化学门控离子通道→后膜对某些离子通透性改变→带电离子发生跨膜流动→后膜发生去极化或超极化→产生突触后电位。

从突触传递冲动的过程可得到以下结论：在突触传递冲动的过程中，突触前末梢去极化是诱发递质释放的关键因素→开启电压门控Ca^{2+}通道；Ca^{2+}是前膜兴奋和递质释放过程的耦联因子→递质释放量与内流入前膜的Ca^{2+}量呈正相关；囊泡膜的再循环利用是突触传递持久进行的必要条件。

突触后电位又分为兴奋性突触后电位和抑制性突触后电位。

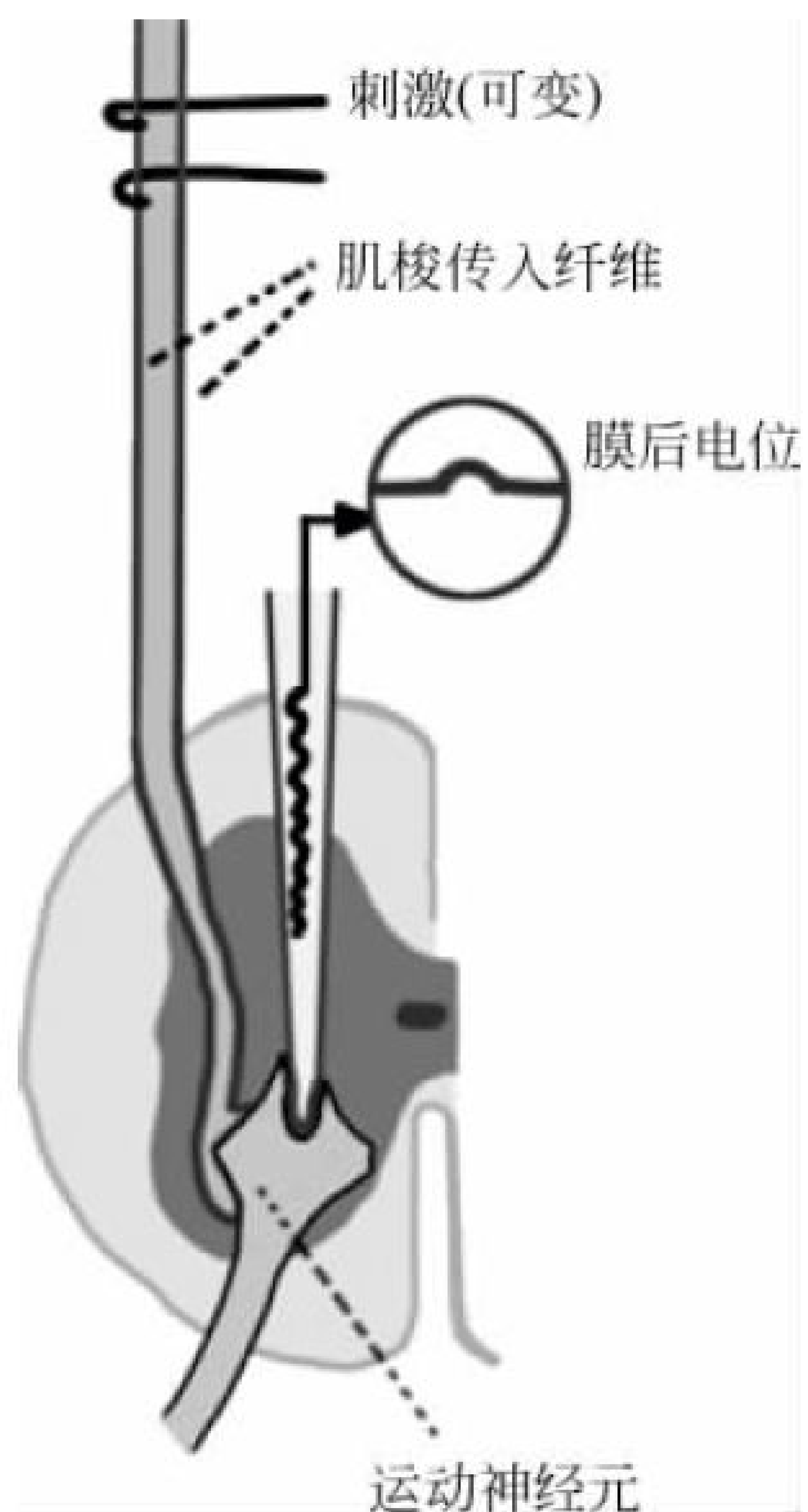


图 6-3 冲动在神经元中的传递

在兴奋性突触后电位的作用下，突触后膜在递质作用下发生去极化，使突触后神经元兴奋性提高。如图 6-3 所示，外部可变刺激作用于肌梭传入纤维后，神经元发生去极

化，产生兴奋性突触后电位。随着刺激强度增加，兴奋性突触后电位发生总和而逐渐增大，使膜电位降低，如使膜电位由静息时的－70mV去极化至－58mV。当兴奋性突触后电位总和达到阈电位(即使膜电位去极化为－52mV)时，系统将冲动传导至整个突触后神经元。

如果在抑制性突触后电位的作用下，突触后膜在递质作用下发生超极化，即如果膜电位静息时为－70mV，超极化后膜电位为－76mV，从而抑制冲动的向后传递。

一个神经元往往与周围的许多神经元形成大量的突触联系，它包含形成众多的兴奋性和抑制性突触，如果兴奋性和抑制性的作用发生在同一个神经元，则将发生整合，即一个神经元最终产生的效应将取决于大量传入信息共同作用的结果。

然而，这种共同的作用不是简单的汇聚作用，因为每一突触形成的位置不同，形成突触后电位的离子流动不同，导致突触传入信息的强度和时间组合的变换足以使神经元接收信息量成倍增加。在突触后膜中，一些突触能够产生大的变换，而另一些可能引起很小的变换。

6.1.3 人工神经网络的定义

神经系统是人体内由神经组织构成的全部装置，主要由神经元组成。神经系统具有重要的功能：一方面它控制与调节各器官、系统的活动，使人体成为一个统一的整体；另一方面通过神经系统的分析与综合，使机体对环境变化的刺激做出相应的反应，达到机体与环境的统一。人的神经系统是亿万年不断进化的结晶，它有着十分完善的“生理结构”和“心理功能”。

因此，以人的大脑组织结构和功能特性为原型设法构建一个与人类大脑结构和功能拓扑对应的人类智能系统是人工神经网络的原则和目标。

1987年Simpson提出了神经网络定义：“人工神经网络是一个非线性的有向图，图中含有可以通过改变权大小来存放模式的加权边，并且可以从不完整的或未知的输入找到模式。”

而在1988年，Hecht-Nielsen提出神经网络的定义：“人工神经网络是一个并行、分布处理结构，它由处理单元及其称为连接的无向信号通道互连而成。这些处理单元(processing element，PE)具有局部内存，并可以完成局部操作。每个处理单元有一个单一的输出连接，这个输出可以根据需要被分支成希望个数的许多并行连接，并且这些并行连接都输出相同的信号，即相应处理单元的信号，信号的大小不因分支的多少而变化。处理单元的输出信号可以是任何需要的数学模型，每个处理单元中进行的操作必须是完全局部的。也就是说，它必须仅仅依赖于经过输入连接到达处理单元的所有输入信号的当前值和存储在处理单元局部内存中的值。”在这一定义中强调：人工神经网络是并行、分布处理结构；一个处理单元的输出可以被任意分支且大小不变；输出信号可以是任意的数学模型；处理单元可以完成局部操作。

目前使用最广泛的是T. Koholen的定义，即“神经网络是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所做出的交互反应”。

6.2 人工神经网络模型

人工神经网络是对人类神经系统的模拟，神经系统以神经元为基础，因此神经网络也是以人工神经元模型为基本构成单位的。

6.2.1 人工神经元的基本模型

今天，计算机科学的分支——联结机制——已经获得相当大的普及。研究领域集中在高度并行计算机架构的行为，也就是说人工神经网络。这些网络使用很多简单计算单元，称为神经元，每一个都试着模拟单个人脑细胞的行为。

神经网络领域的研究者已经分析了人类脑细胞的不同模型。人脑包含约140亿个神经细胞。如图6-4为人类神经元的简化原理图。

细胞本身包含的细胞核被电气膜包围。每一个神经元有一个激活水平，其范围在最大值与最小值之间。因此，与布尔逻辑相比，不仅仅是两个可能值或可能存在的状态。

突触存在增加或减少这个神经元的激活程度，作为其他神经元的输入结果。这些突触从一个发送的神经元到一个接收神经元传输激活水平，如果突触是兴奋的，发送神经元的激活水平增加接收神经元的激活水平；如果突触是抑制的，发送神经元的激活水平减少接收神经元的激活水平。突触差异不仅仅在于它们是否兴奋或抑制接收神经元，也在于影响的权值(突触强度)。每个神经元的输出都由轴突转换。

综上所述，生物神经元信息传递的过程是：当一个兴奋性的冲动到达突触前膜持续约0.5ms，其去极性效应就会在突触后膜上记录下来，随着突触后膜接触的神经递质量增加而增加其幅度，并增加突出后神经元对刺激的兴奋性反应；与此相反，抑制性突触后电位可使突触后神经元对后继刺激的兴奋性反应降低，兴奋性突触后电位与抑制性突触后电位在时空上可进行代数累积，一旦这种累积超过某个阈值，神经元即发生动作电位或神经冲动。

如果将上述过程用数学图形方式表示，则可获得人工神经元的模型，如图6-5所示。

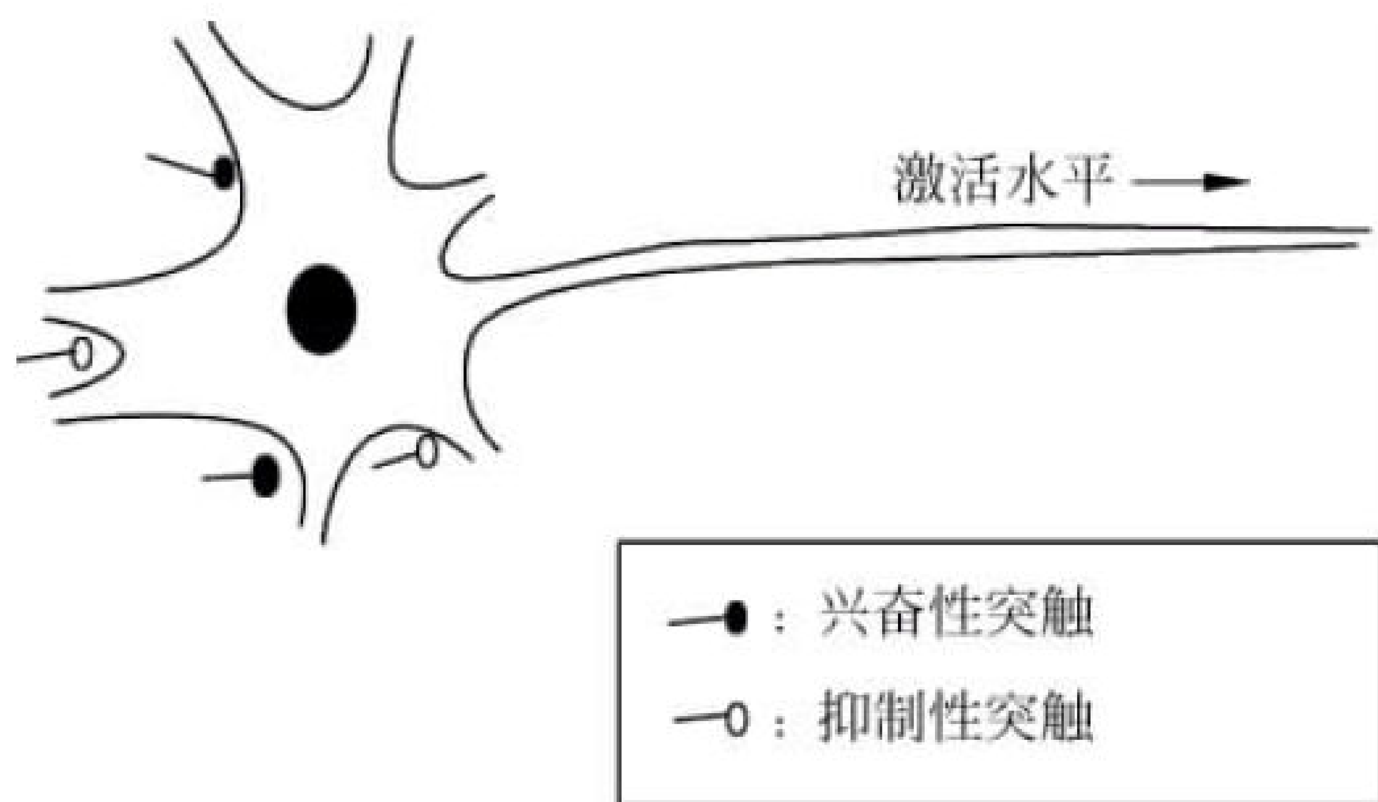


图6-4 人类神经元的简化原理图

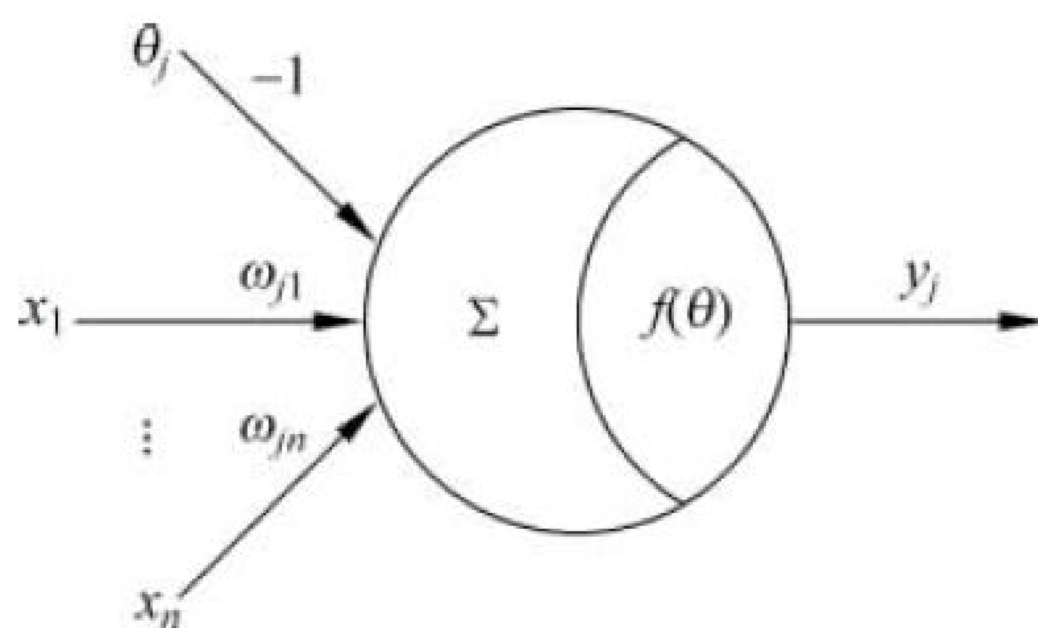


图6-5 人工神经元模型

人工神经元模型为一个多输入单输出的信息处理单元。其中，ω_{ji}为输入信号加权值；θ为阈值，即输入信号的加权乘积的和必须大于阈值，输入信号才能向后传递；$f(\theta)$为输入

信号与输出信号的转换函数。常见的转换函数如图 6-6 所示。

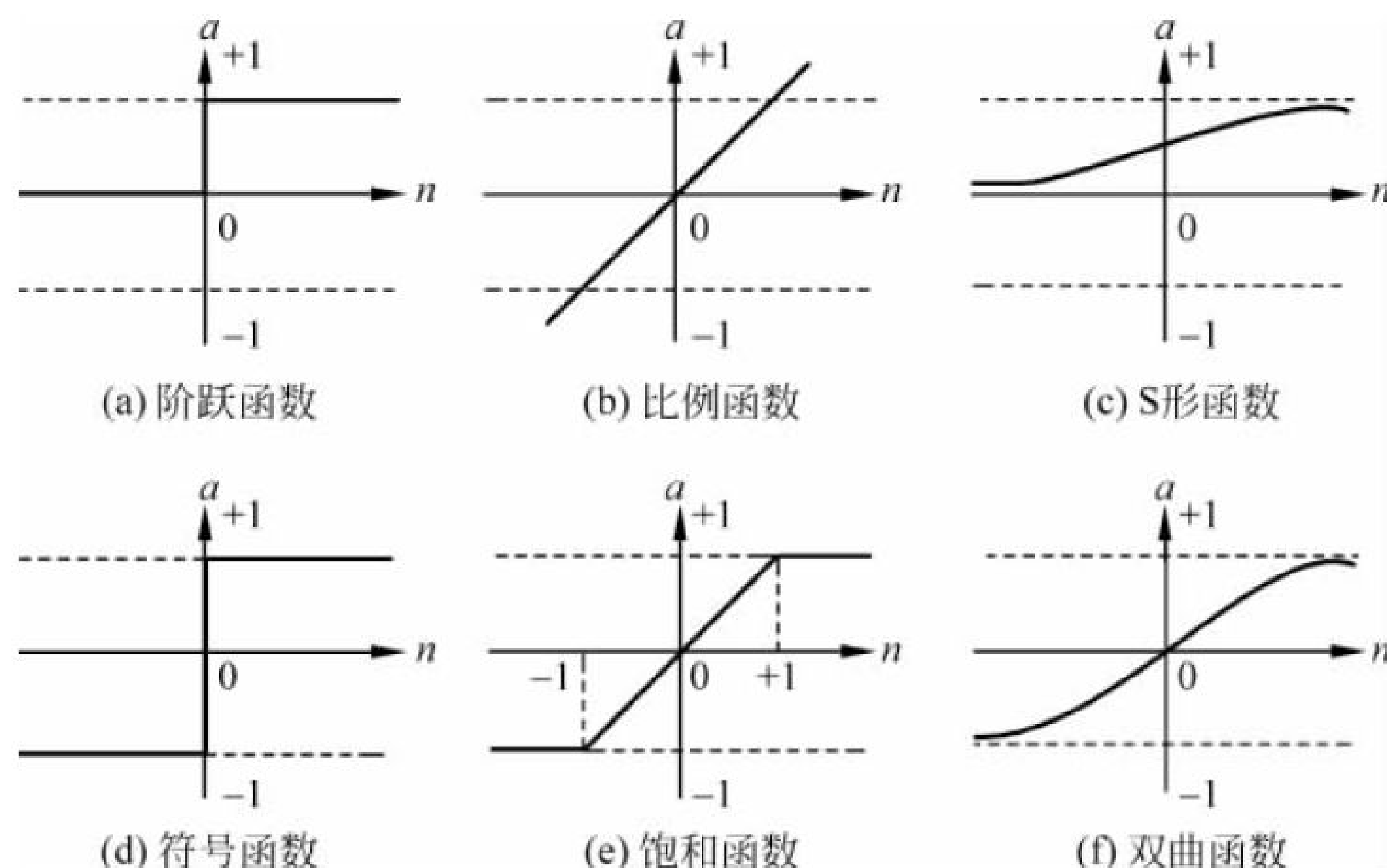


图 6-6 常见的转换函数

阶跃函数的解析表达式：$a=f(n)=\begin{cases}1, & n\geqslant 0\\ 0, & n<0\end{cases}$ (6-1)

比例函数的解析表达式：$a=f(n)=n$ (6-2)

S 形函数的解析表达式：$a=f(n)=\dfrac{1}{1+\mathrm{e}^{-\mu n}}$ (6-3)

符号函数的解析表达式：$a=f(n)=\begin{cases}1, & n\geqslant 0\\ -1, & n<0\end{cases}$ (6-4)

饱和函数的解析表达式：$a=f(n)=\begin{cases}1, & n\geqslant 1\\ n, & -1<n<1\\ -1, & n<-1\end{cases}$ (6-5)

双曲函数的解析表达式：$a=f(n)=\dfrac{1-\mathrm{e}^{-\mu n}}{1+\mathrm{e}^{-\mu n}}$ (6-6)

6.2.2 人工神经网络基本构架

人脑之所以有高等智慧能力是因为有大量的生物神经细胞构成的神经网络。同样，若要让“人工神经网络”具有一定程度人的智慧，则必须将许多的人工神经元经由适当的连接，构架一个“类神经网络”的网络，我们称这一“类神经网络”为人工神经网络。

一个神经网络包括一组交互连接的同样单元。每个单元可能被看作从许多其他单元聚合信息的简单处理器。聚合后，这个单元计算通过通路连接到其他单元的输出。一些单元通过输入单元或输出单元被连接到外部世界。信息通过输入单元首先传入系统，接着通过网络处理并从输出单元读取。

基于简单神经元模型，存在不同的数学模型。图 6-7 为人工神经元的基本结构。

单个神经元的行为由下面的函数确定。

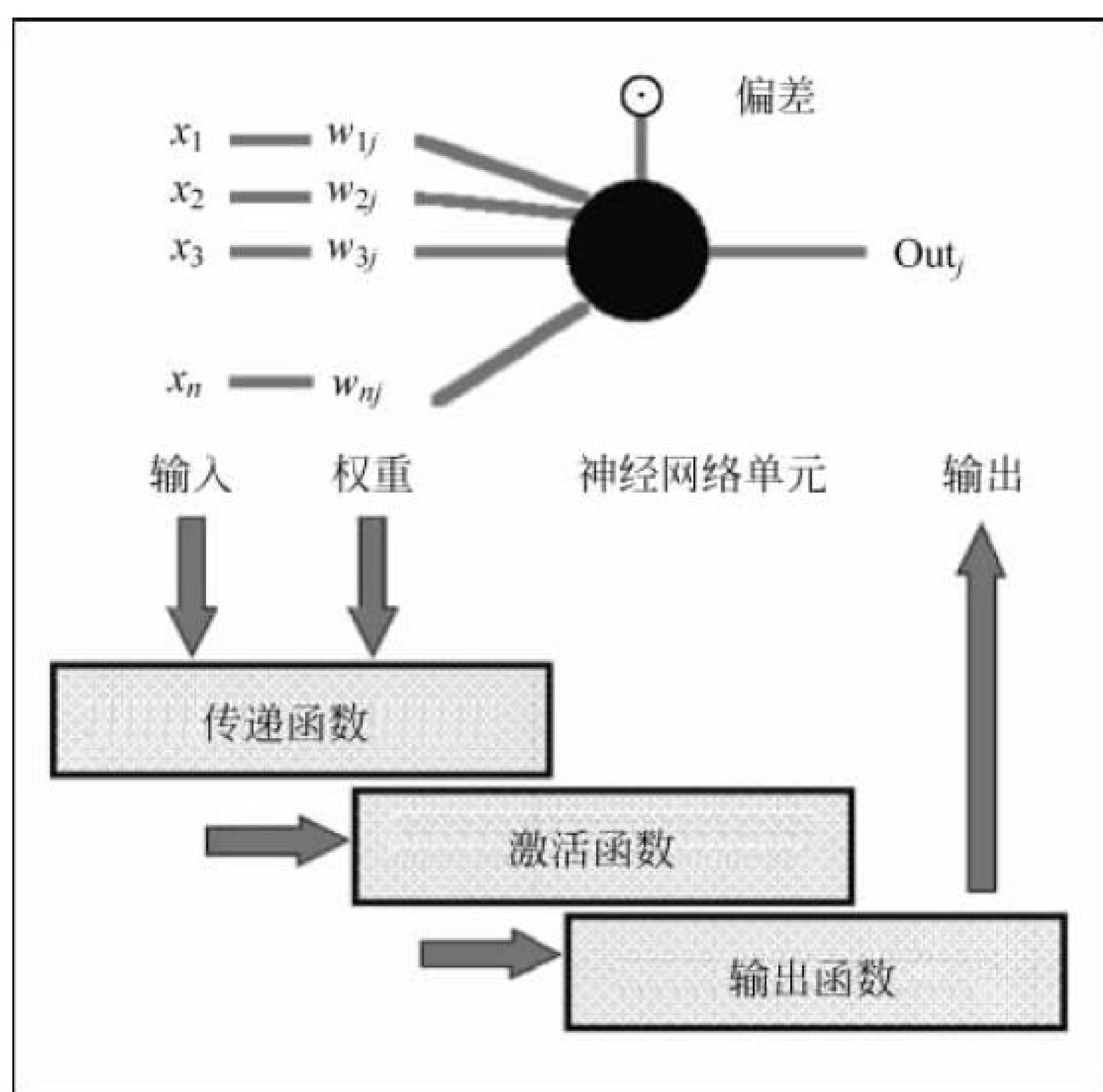


图 6-7　人工神经元的基本结构

1. 传播函数

首先，传播函数结合所有基于发送神经元的输入 x_i。组合的方法主要是加权和，权 w_i 代表突触的强度。刺激突触为正的权值；抑制突触为负的权值。偏差 θ 被加到加权和，表达神经元的后台激活水平。

2. 激活函数

传播函数的结果现在用于计算有所谓激活函数的神经元的激活。不同类型的函数用于这一函数计算，其中 S 形函数是最常用的。

3. 输出函数

有时，由激活函数产生的计算结果要接着被其他输出函数进一步处理，这将允许额外过滤每个单元的输出信息。

就是这样简单的神经元模型支撑着今天大多数的神经网络应用。

注意：这个模型仅是实际神经网络的一个很简单的近似描述。目前为止还不能准确地建立一个单个的人类神经元模型，因为建模已超出了人类当前的技术能力。因此，基于这个简单神经元模型的任何应用都不能准确复制人脑。但是，很多成功应用这个技术的例子证明，基于简单神经元模型的神经网络具有一定的优点。

从上面的结构可知，人工神经网络用于模拟生物神经网络。模拟从以下两个方面进行：一是从结构和实现机理方面进行模拟；二是从功能上进行模拟。根据不同的应用背景及不同的应用要求，实际的神经网络结构形式多样，其中最典型的人工神经网络结构如图 6-8 所示。

前馈型网络是一类单方向层次性网络模块，它包含输入层、中间隐含层和输入层。每一

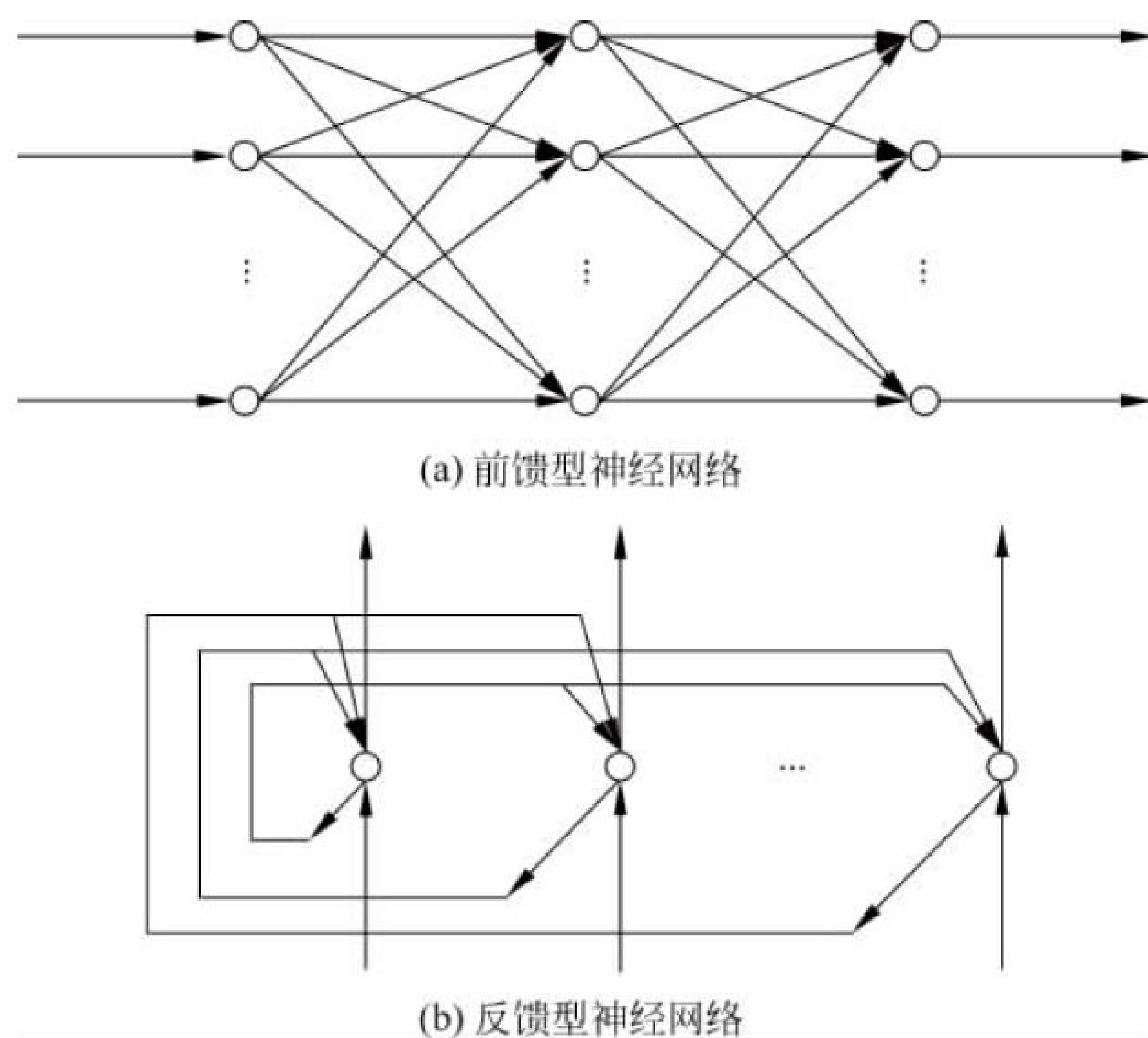


图 6-8 最典型的人工神经网络结构

层皆由一些神经元构架而成,而同一层中的神经元彼此不相连,不同层间的神经元则彼此相连。信号的传输方向也是单方向的,由输入层传输至输出层。这种类型的网络结构简单,可实现反应式的感知、识别和推理。

反馈型网络是一类可实现联想记忆即联想映射的网络。网络中的人工神经元彼此相连,对每个神经元而言,它的输出连接至所有其他神经元,而它的输入则来自所有其他神经元的输出。可以说,网络中的每个神经元平行地接受所有神经元输入,再平行地将结果输出到网络中其他神经元上。反馈型神经网络在智能模拟中得到广泛应用。

从人工神经网络的结构可知,人工神经网络是一个并行和分布式的信息处理网络,由多个神经元组成,每一个神经元有一个单一的输出,可以连接到很多其他的神经元,输入有多个连接通路,每个连接通路对应一个连接权系数。

6.2.3 人工神经网络的工作过程

就像人的认知过程一样,人工神经网络也存在学习的过程。在神经网络结构图中,信号的传递过程中要不断进行加权处理,即确定系统各个输入对系统性能的影响程度,这些加权值是通过对系统样本数据的学习确定的。当给定神经网络一组已知的知识,在特定的输入信号作用下,反复运算网络中的连接权值,使其得到期望的输出结果,这一过程称为学习过程。

对于前馈型神经网络,它从样本数据中取得训练样本及目标输出值,然后将这些训练样本当作网络的输入,利用最速下降法反复地调整网络的连接加权值,使网络的实际输出与目标输出值一致。当输入一个非样本数据时,已学习的神经网络就可以给出系统最可能的输出值。

对于反馈型神经网络,它从样本数据中取得需记忆的样本,并以 Hebbian 学习规则来

调整网络中的联结加权值，以“记忆”这些样本。当网络将样本数据记忆完成，这时如果给神经网络一个输入，当这一输入是一个“不完整”“带有噪声”的数据时，神经网络通过联想，将输入信号与记忆中的样本对照，给出输入所对应的最接近的样本数据的输出值。

6.2.4　人工神经网络的特点

基于神经元构建的人工神经网络具有如下特点。

1. 并行数据处理

人工神经网络采用大量并行计算方式，经由不同的人工神经元进行运算处理。因此，用硬件实现的神经网络的处理速度远远高于通常计算机的处理速度。

2. 强的容错能力

人工神经网络在运作时具有很强的容错能力，即使输入信号“不完整”或“带有噪声”，也不会影响其运作的正确性。而且即使有部分人工神经元损坏，也不会影响整个神经网络的整体性能。

3. 具有泛化能力

通过记忆已知样本数据，对其他输入信号进行运算，计算该输入相对应的输出值。

4. 可实现最优化计算

神经网络可在约束条件下，使整个设计目标达到最优化状态。

5. 具有自适应功能

神经网络可以根据系统提供的样本数据，通过学习和训练，找出和输出之间的内在联系，从而求得问题的解，而不是依赖对问题的经验知识和规则，因此它具有很好的适应性。

6.3　前馈神经网络

对于很多应用，一个确定的网络计算与确定的时间行为一样重要。网络架构允许中间单元的循环结构计算依靠神经元内部激活的输出值。即使输入不变化，输出也可能不同，直到网络内的计算达到稳定状态。单元之间不仅有单方向连接的网络，而且有反方向的网络，这些相同方向的网络称为前馈网络。在实际应用中，前馈类型的网络非常重要。

神经网络的对象以之前训练得到的网络处理信息，使用输入和相应输出样本数据集，或者估计神经网络性能的“教师”进行网络训练。神经网络使用学习算法完成期望的训练。之前建立的神经网络未经训练，不能反映任何行为。学习算法连接着网络进行训练，并修改网络的单个神经元和它们连接的权，使网络行为反映期望的行为。网络学习的知识通常由连接单元的连接强度表示，有时也由单元自己的配置表示。

所以,用户如何使一个神经网络学习呢?方法类似于巴甫洛夫对狗的训练。几百年前,研究者巴甫洛夫使用狗进行实验。当他拿出狗食时,狗在流口水。他接着在狗笼装了一个铃。当他敲铃时狗没有流口水,因此他看到在铃和食物之间没有联系。他接着使用铃声训练狗使其和食物发生联系,每当他拿出狗食时总是让铃响。一段时间后,当铃声响起没有食物时,狗也流口水。巴甫洛夫的实验原理如图6-9所示。



图6-9　巴甫洛夫的实验原理

假设将简单神经元模型看作是巴甫洛夫的狗。有两个输入神经元,一个表示狗看食物的事实;另一个表示铃响的事实。输入神经元与输出神经元的连接叫作突触。线的虚实表示突触的权。在学习之前,狗仅对食物有反应,对铃声无反应。因此,从左边输入神经元到输出神经元的线是实的,而从右边输入神经元到输出神经元的线是虚的。

之后,当给出食物时,重复让狗在铃和食物间建立关联。因此,右边的线也变实——突触的权增加。从这些实验中研究者使用Hebb名字演绎出下面的学习规则:

如果神经元的输出要激活,增加活性输入神经元的权;

如果神经元的输出要停止,减小活性输入神经元的权。

这个规则叫做Hebbian规则,是所有学习算法之父。必须关注学习原理以说明这个规则如何被应用到今天的学习方法中。

1）监督学习

如果一个给定的输入模式(如一个可识别字符)必须与指定的输出模式关联(如所有有效字符集),则可以通过对照计算的结果和期望的结果监督彼此学习。

2）非监督学习

如果训练过程的任务要发现环境的规律性(像给定输入模式的类属性),通常没有指定的输出模式或结构监督训练结果,这个学习过程被叫作非监督学习。

神经网络行为通过改变连接单元的连接强度配置输出。监督学习仅能使用与这个过程完全独立的样本数据完成。因此,学习和工作阶段不能分割。

1）训练阶段

建立一个神经网络解决方案意味着要训练网络按照期望的行为运行,这一训练过程称为学习阶段。样本数据集或“教师”在这一阶段使用。“教师”也是一个数学函数或估计神经网络性能质量的人。因为神经网络多用于没有适当数学模型存在的复杂应用,并且神经网络性能在大多数应用中很难评估,所以大多数系统使用样本数据训练。

2）工作阶段

学习完成后,神经网络准备进入工作阶段。作为一个训练结果,当输入值匹配训练样本之一时,神经网络输出值几乎等于样本数据集的那些值。对于样本数据输入值中间的输入

值,近似输出值。在工作阶段,神经网络的行为是确定的。因此,每个可能输入值的组合总是产生同样的输出值。在工作阶段,神经网络不能学习,这在大多数技术应用中是重要的,它是确保系统永远也不要走向危险状态的前提。

监督学习的训练阶段和工作阶段示意图如图 6-10 所示。

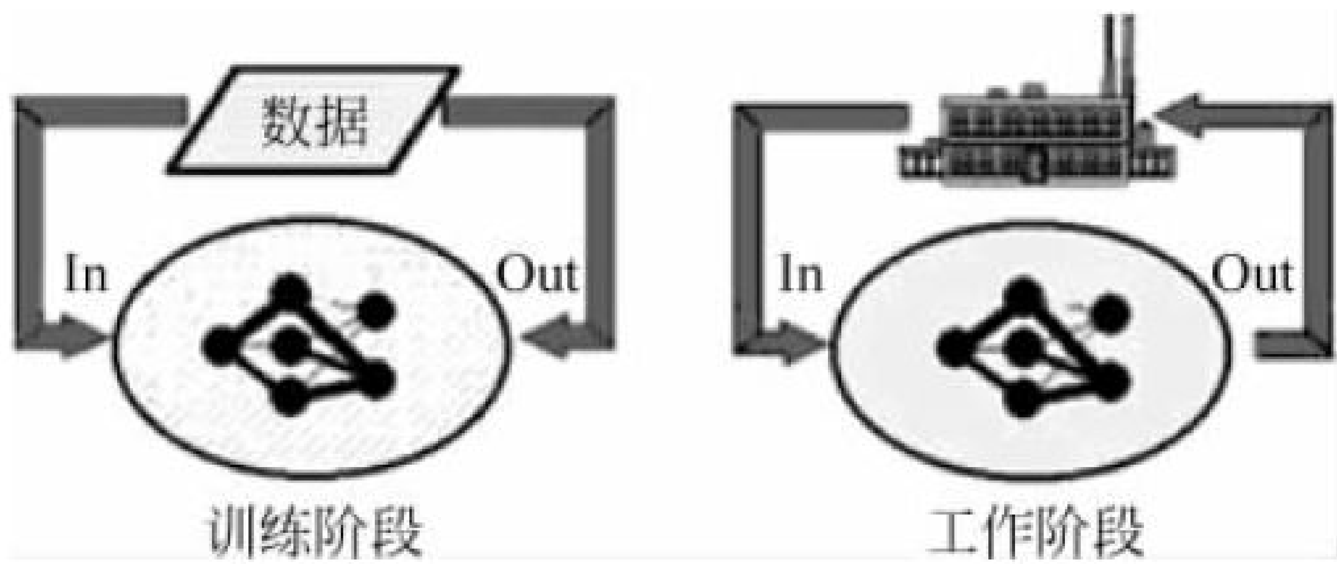


图 6-10 监督学习的训练阶段和工作阶段示意图

今天,经常使用的神经网络算法是将监督训练程序应用到前馈网络中。

前馈型神经网络具有分层结构,第一层是输入层,中间是隐含层,最后一层是输出层。其信息从输入层依次向后传递,直至输出层。

6.3.1 感知器网络

感知器网络是最简单的前馈网络,主要用于模式分类,也可用在基于模式分类的学习控制和多模态控制中。

1. 单层感知器网络

单层感知器网络结构如图 6-11 所示。图中 $\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$ 是输入特征向量,ω_{ij} 是 x_i 到 y_j 的连接权,输出量 $y_j(j=1,2,\cdots,n)$是按照不同特征的分类结果。由于按不同特征的分类是互相独立的,因此可以取出其中的一个神经元来讨论,如图 6-12 所示。

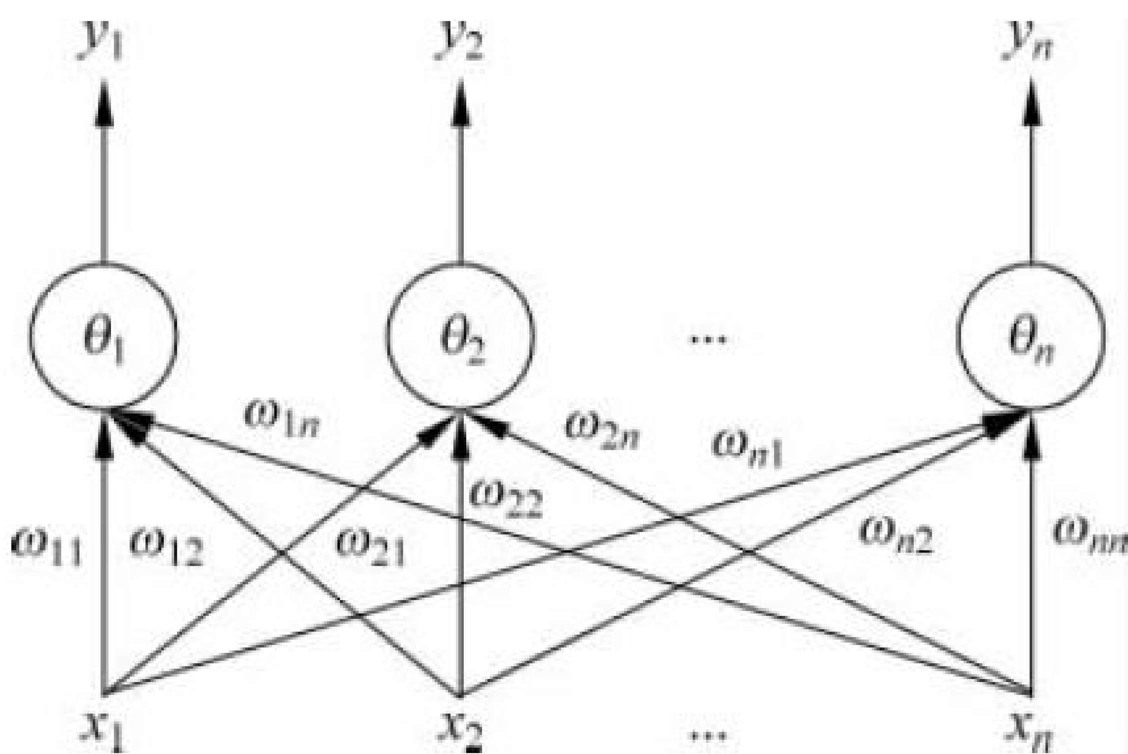


图 6-11 单层感知器网络结构

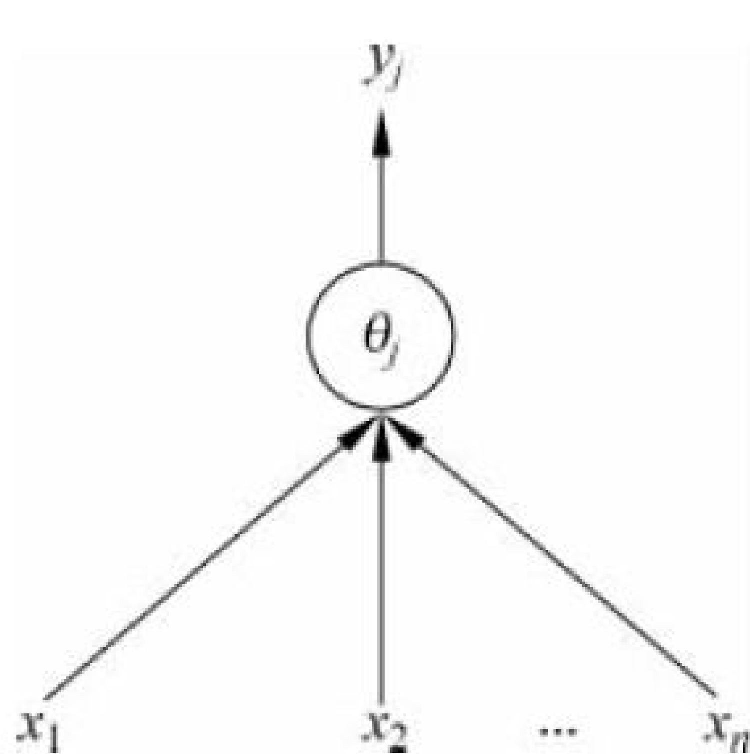


图 6-12 单个神经元的感知器

其输入到输出的变换关系为

$$s_j = \sum_{i=1}^{n} \omega_{ij} x_i - \theta_j \tag{6-7}$$

$$y_j = f(s_j) = \begin{cases} 1, & s_j \geqslant 0 \\ -1, & s_j < 0 \end{cases} \tag{6-8}$$

若有 P 个输入样本 $x^p(p=1,2,\cdots,P)$，经过该感知器的输出，y_j 只有两种可能，即 $y_j=1$ 或 $y_j=-1$，从而说明它将输入模式分成了两类。若将 $x^p(p=1,2,\cdots,P)$ 看成是 n 维空间的 P 个点，则该感知器将该 P 个点分成了两类，它们分属于 n 维空间的两个不同的部分。

以二维空间为例，如图 6-13 所示。图中以三角形和长方形代表输入的特征点，三角形和长方形表示具有不同特征的两类向量。根据感知器的变换关系，可知分界线的方程为

$$\omega_1 x + \omega_2 y - \theta = 0 \tag{6-9}$$

显然，这是一条直线方程，它说明只有那些线性可分模式类才能用感知器来加以区分。图 6-14 所示的异或关系，显然是线性不可分的。因此，单层感知器网络对异或关系的两维输入是线性不可分的。

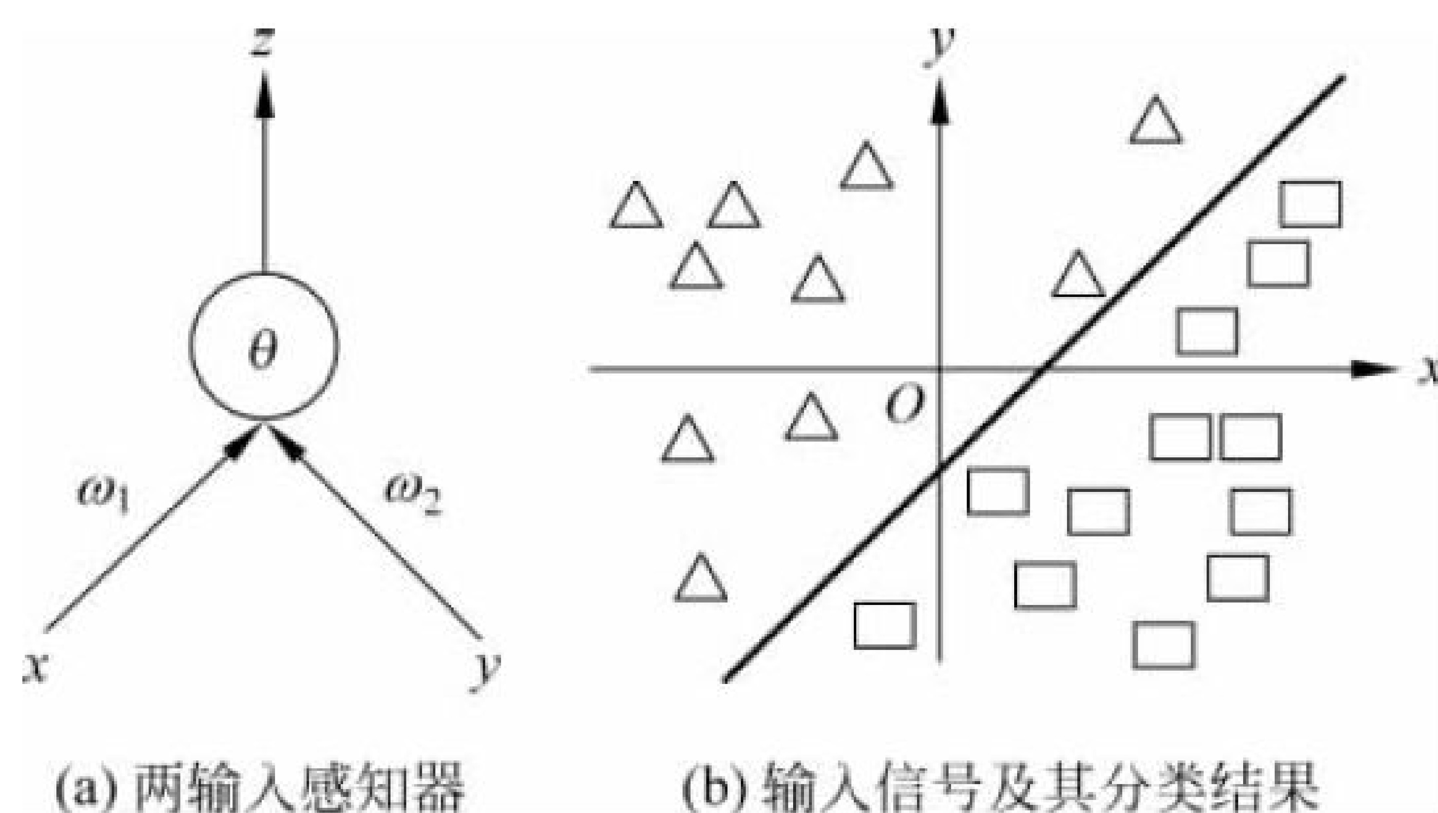


图 6-13　二维输入的感知器网络

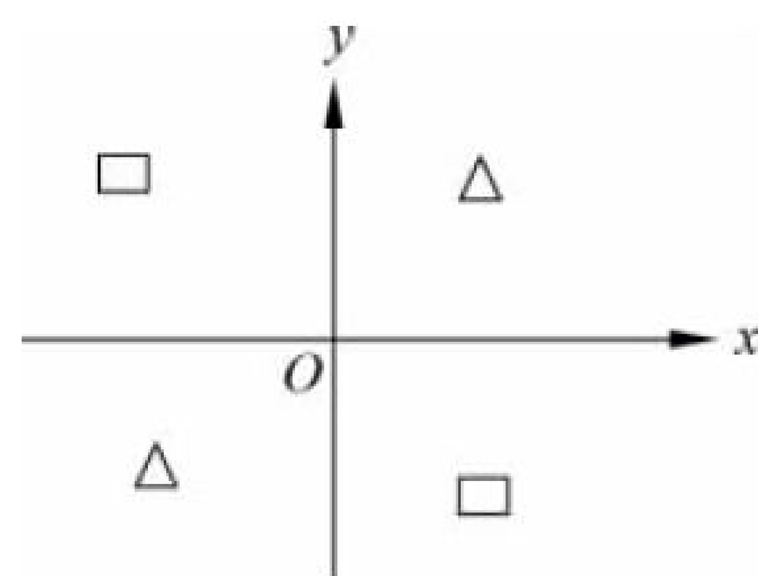


图 6-14　二维输入信号为异或关系

2. 多层感知器网络

由于不可能对单层感知器网络实现正确区分，因此需要增加神经元数量。对于上例中提到的异或问题，可采用图 6-15 所示的两层二维输入的感知器网络实现异或逻辑。

(1) 第一层第一个神经元所完成的工作为 $\omega_{11}x+\omega_{12}y-\theta_1^1=0$，即在输入点坐标中产生第 1 条分类线，如图 6-16 所示。

(2) 第一层第二个神经元所完成的工作为 $\omega_{21}x+\omega_{22}y-\theta_2^1=0$，即在输入点坐标中产生第 2 条分类线，如图 6-17 所示。

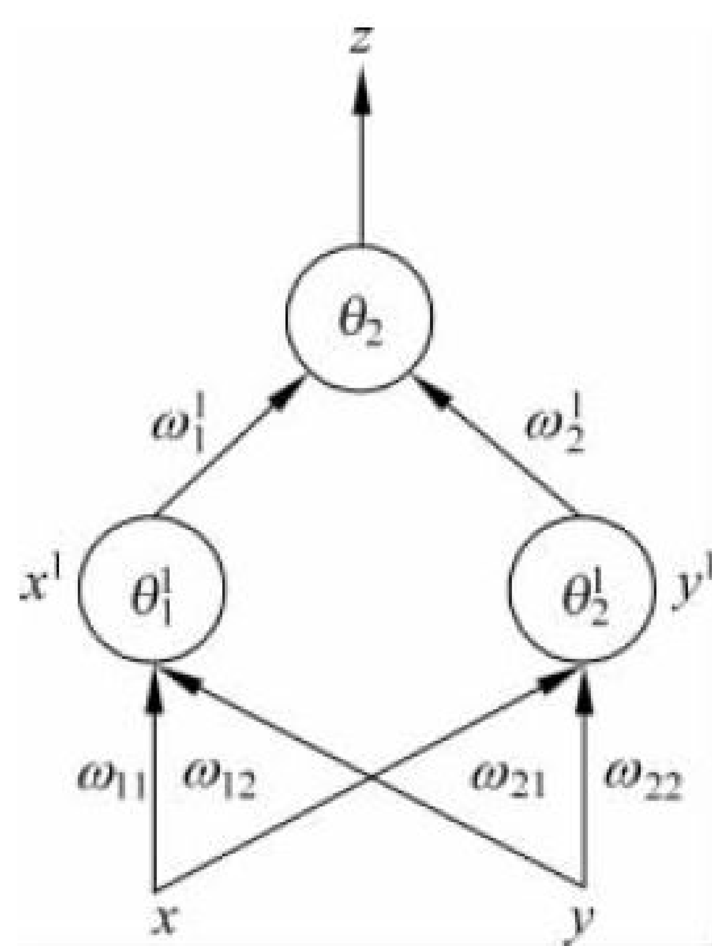


图 6-15　两层二维输入的感知器网络

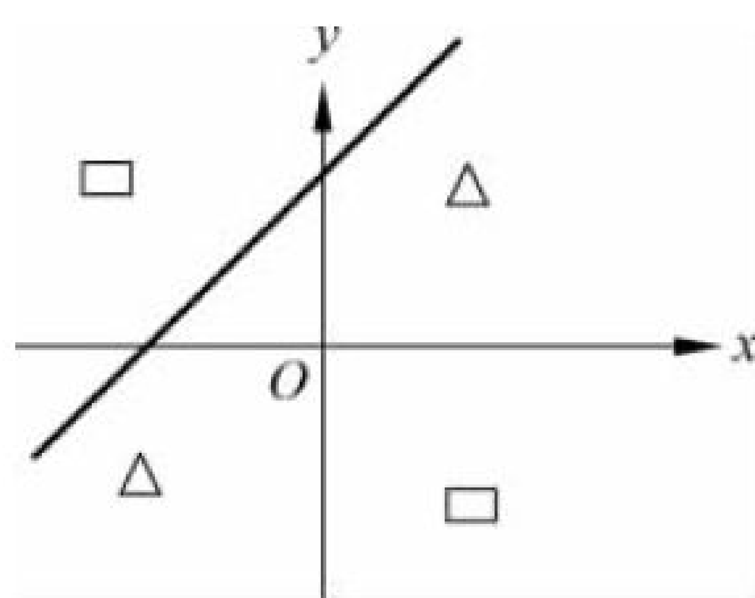


图 6-16　异或关系第一次分类

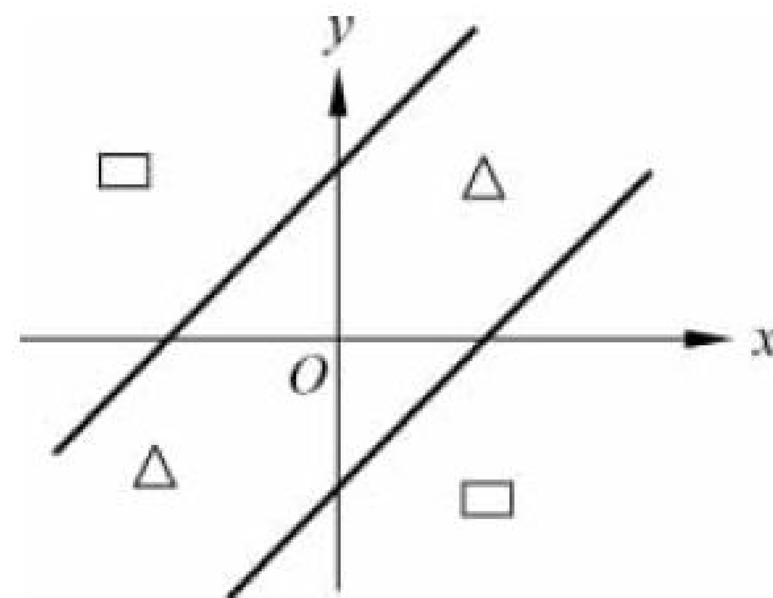


图 6-17　异或关系第二次分类

(3) 第二层神经元所完成的工作为 $\omega_1^1 x^1 + \omega_2^1 y^1 - \theta_2 = 0$,即将上述两条直线所确定的区域进行划分,从而将具有异或关系的输入进行分类。

从上述对异或输入的处理可知,只要建立足够多的神经元连接,即构建多层感知器网络,就可以实现任意形状的划分。多层感知器网络的结构如图 6-18 所示,第一层为输入层,中间层为隐含层,最后一层为输出层。

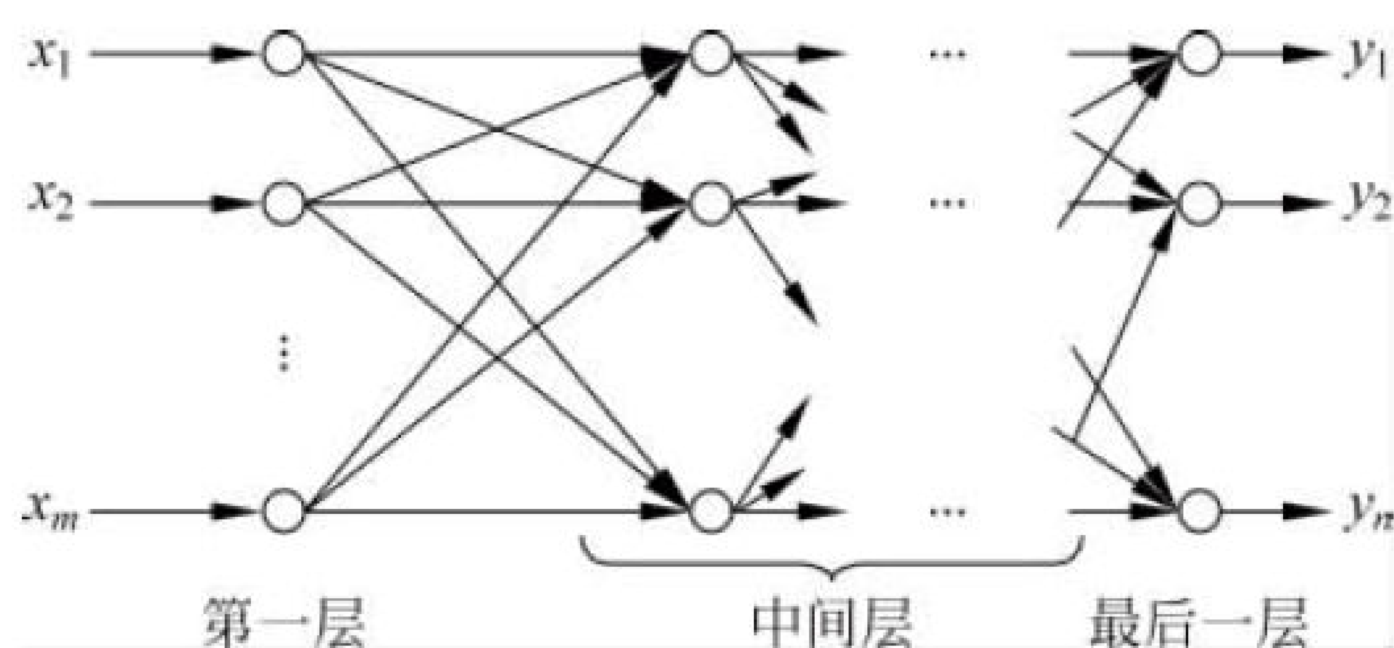


图 6-18 多层感知器网络的结构图

6.3.2 BP 网络

感知器网络中,神经元的变换函数采用的是符号函数,即输出为二值量 1 或 −1,它主要用于模式分类。当神经元变换函数采用 S 形函数时,系统的输出量将为 0～1 的连续量,它可实现从输入到输出的任意非线性映射。由于连接权的调整采用的是反向传播(back propagation,BP)的学习算法,因此该网络也称为 BP 网络。

反向传播网络是将 W-H 学习规则一般化,对非线性可微分函数进行权值训练的多层网络,权值的调整采用反向传播的学习算法。其主要思想是从后向前(反向)逐层传播输出层的误差,以间接计算出隐含层误差。算法分为两部分,第一部分(正向传播过程)输入信息从输入层经隐含层逐层计算各单元的输出值;第二部分(反向传播过程)输出误差逐层向前计算出隐含层各单元的误差,并用此误差修正前层权值。

反向传播包含两个过程,即正向传播和反向传播。

(1) 正向传播:输入的样本从输入层经过隐含层单元一层一层进行处理,通过所有的隐含层之后,则传向输出层;在逐层处理的过程中,每一层神经元的状态只对下一层神经元的状态产生影响。在输出层把当前输出和期望输出进行比较,如果当前输出不等于期望输出,则进入反向传播过程。

(2) 反向传播:把误差信号按照原来正向传播的通路反向传回,并对每个隐含层的各个神经元的连接权系统进行调整,以使期望误差信号趋于最小。

BP 网络的计算过程如下:设第 q 层($q=1,2,\cdots,Q$)的神经元个数为 n_q,输入到第 q 层的第 i 个神经元的连接权系数为 ω_{ij}^q($i=1,2,\cdots,n_q$; $j=1,2,\cdots,n_{q-1}$),则该多层感知器网络的输入/输出变换关系为

$$s_i^q = \sum_{j=0}^{n_{q-1}} \omega_{ij}^q x_j^{q-1} \tag{6-10}$$

其中,$\omega_{i0}^q = -1$; $x_0^{q-1} = \theta_i^q$; $x_i^{q-1} = f(s_i^q) = \dfrac{1}{1+\mathrm{e}^{-\mu s_i^q}}$; $i=1,2,\cdots,n_q$; $j=1,2,\cdots,n_{q-1}$; $q=1,$

$2,\cdots,Q$。

设给定 P 组输入/输出样本 $x_p^0=[x_{p1}^0,x_{p2}^0,\cdots,x_{pn_0}^0]^{\mathrm{T}},d_p=[d_{p1},d_{p2},\cdots,d_{pn_Q}]^{\mathrm{T}}(p=1,2,\cdots,P)$,利用该样本集首先对 BP 网络进行训练,即对网络的连接权系数进行学习和调整,以使该网络实现给定的输入/输出映射关系。经过训练的 BP 网络,对于不是样本集中的输入也能给出合适的输出。该性质称为泛化(generalization)功能。从函数拟合的角度看,说明 BP 网络具有插值功能。

对于 BP 神经网络,设取拟合误差函数的代价函数为

$$E=\frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{n_Q}(d_{pi}-x_{pi}^Q)^2=\sum_{p=1}^{P}E_p \tag{6-11}$$

即

$$E_p=\sum_{i=1}^{n_Q}(d_{pi}-x_{pi}^Q)^2 \tag{6-12}$$

问题是如何调整连接权系数以使代价函数 E 最小。优化计算的方法很多,比较典型的是一阶梯度法,即最速下降法。

一阶梯度法寻优的关键是计算优化目标函数(即本问题中的误差代价函数)E 对寻优参数的一阶导数。即

$$\frac{\partial E}{\partial \omega_{ij}^q},\quad q=Q,Q-1,\cdots,1 \tag{6-13}$$

由于 $\dfrac{\partial E}{\partial \omega_{ij}^q}=\sum\limits_{p=1}^{P}\dfrac{\partial E_p}{\partial \omega_{ij}^q}$,因此下面重点讨论$\dfrac{\partial E_p}{\partial \omega_{ij}^q}$的计算。

对于第 Q 层,有

$$\frac{\partial E_p}{\partial \omega_{ij}^Q}=\frac{\partial E_p}{\partial x_{pi}^Q}\frac{\partial x_{pi}^Q}{\partial s_{pi}^Q}\frac{\partial s_{pi}^Q}{\partial \omega_{ij}^Q}=-(d_{pi}-x_{pi}^Q)f'(s_{pi}^Q)x_{pi}^{Q-1}=-\delta_{pi}^Q x_{pi}^{Q-1} \tag{6-14}$$

其中

$$\delta_{pi}^Q=-\frac{\partial E_p}{\partial x_{pi}^Q}=(d_{pi}-x_{pi}^Q)f'(s_{pi}^Q) \tag{6-15}$$

x_{pi}^Q、s_{pi}^Q 及 x_{pi}^{Q-1} 表示利用第 p 组输入样本所算得的结果。

对于第$(Q-1)$层,有

$$\frac{\partial E_p}{\partial \omega_{ij}^{Q-1}}=\frac{\partial E_p}{\partial x_{pi}^{Q-1}}\frac{\partial x_{pi}^{Q-1}}{\partial \omega_{ij}^{Q-1}}=\left(\sum_{k=1}^{n_Q}\frac{\partial E}{\partial s_{pk}^Q}\frac{\partial s_{pk}^Q}{\partial x_{pi}^{Q-1}}\right)\frac{\partial x_{pi}^{Q-1}}{\partial s_{pk}^{Q-1}}\frac{\partial s_{pk}^{Q-1}}{\partial \omega_{ij}^{Q-1}} \tag{6-16}$$

$$=\left(\sum_{k=1}^{n_Q}-\delta_{pk}^Q\omega_{ki}^Q\right)f'(s_{pi}^{Q-1})x_{pj}^{Q-2}=-\delta_{pi}^{Q-1}x_{pj}^{Q-2} \tag{6-17}$$

其中

$$\delta_{pi}^{Q-1}=-\frac{\partial E_p}{\partial x_{pi}^{Q-1}}=\left(\sum_{k=1}^{n_Q}\delta_{pk}^Q\omega_{ki}^Q\right)f'(s_{pi}^{q-1}) \tag{6-18}$$

显然,它是反向递推计算的公式,即首先计算出 δ_{pi}^Q,然后递推计算出 δ_{pi}^{Q-1}。依此类推,可继续反向递推计算出 δ_{pi}^q 和$\dfrac{\partial E_p}{\partial \omega_{ij}^{q1}}$,$q=Q\times 2,Q\times 3,\cdots,1$。从式中可以看出,在 δ_{pi}^q 的表达式中包含了导数项 $f'(s_{pi}^q)$,由于假定 $f(\cdot)$为 S 形函数,因此可求得其导数

$$x_{pi}^q=f(s_{pi}^q)=\frac{1}{1+\mathrm{e}^{-\mu s_{pi}^q}} \tag{6-19}$$

$$f'(s_{pi}^q) = \frac{\mu e^{-\mu s_{pi}^q}}{(1+e^{-\mu s_{pi}^q})^2} = \mu f(s_{pi}^q)[1-f(s_{pi}^q)] = \mu x_{pi}^q(1-x_{pi}^q) \tag{6-20}$$

最后可归纳出 BP 网络的学习算法如下

$$\omega_{ij}^q(k+1) = \omega_{ij}^q(k) + \alpha D_{ij}^q(k), \quad \alpha > 0 \tag{6-21}$$

$$D_{ij}^q = \sum_{p=1}^{P} \delta_{pi}^q x_{pj}^{q-1} \tag{6-22}$$

$$\delta_{pi}^q = \left(\sum_{k=1}^{n_Q-1} \delta_{pk}^{q+1} \omega_{ki}^{q+1}\right) \mu x_{pi}^q (1-x_{pi}^q) \tag{6-23}$$

$$\delta_{pi}^Q = (d_{pi} - x_{pi}^Q) \mu x_{pi}^Q (1-x_{pi}^Q) \tag{6-24}$$

其中，$q=Q,Q-1,\cdots,1$；$i=1,2,\cdots,n_q$；$j=1,2,\cdots,n_{q-1}$。

对于给定的样本集，目标函数 E 是全体连接权系数 ω_{ij}^q 的函数。因此，要寻优的参数 ω_{ij}^q 个数比较多。也就是说，目标函数 E 是关于连接权的一个非常复杂的超曲面，这就给寻优带来一系列问题。其中最大的一个问题就是收敛速度慢。由于待寻优的参数太多，必然导致收敛速度慢的缺点。第二个问题就是系统可能陷入局部极值，即 E 的超曲面可能存在多个极值点。按照上面的寻优算法，它一般收敛到初值附近的局部极值。

BP 网络具有以下主要优点：

(1) 只有有足够多的隐含层节点和隐含层，BP 网络才可以逼近任意的非线性映射关系。

(2) BP 网络的学习算法属于局部逼近的方法，因此它具有较好的泛化能力。

BP 网络的主要缺点如下：

(1) 收敛速度慢。

(2) 容易陷入局部极值点。

(3) 难以确定隐含层和隐含层节点的个数。

由于 BP 网络有很好的逼近非线性映射的能力，因此它可应用于信息处理、图像识别、模型辨识、系统控制等方面。

6.3.3　BP 网络的建立及执行

1. 建立 BP 网络

首先需要选择网络的层数和每层的节点数。

对于具体问题，若确定了输入变量和输出变量，则网络输入层和输出层的节点个数与输入变量个数及输出变量个数对应。隐含层节点的选择应遵循以下原则：在能正确反映输入/输出关系的基础上，尽量选取较少的隐含层节点，而使网络尽量简单。一种方法是先设置较少的节点，对网络进行训练，并测试网络的逼近能力，然后逐渐增加节点数，直到测试的误差不再有明显的较小为止；另一种方法是先设置较多的节点，在对网络进行训练时，采用如下的误差代价函数

$$E_f = \frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{n_Q}(d_{pi}-x_{pi}^Q)^2 + \varepsilon\sum_{q=1}^{Q}\sum_{i=1}^{n_Q}\sum_{j=1}^{n_Q-1}|\omega_{ij}^q| = E + \varepsilon\sum_{q,i,j}|\omega_{ij}^q| \tag{6-25}$$

其中,E 仍与以前的定义相同,它表示输出误差的平方和。第二项的作用相当于引入一个“遗忘”项,其目的是为了使训练后的连接权系数尽量小。可以求得这时 E_f 对 ω_{ij}^q 的梯度为

$$\frac{\partial E_f}{\partial \omega_{ij}^q}=\frac{\partial E}{\partial \omega_{ij}^q}+\varepsilon \operatorname{sgn}(\omega_{ij}^q) \tag{6-26}$$

利用该梯度可以求得相应的学习算法。在训练过程中只有那些确实有必要的连接权才予以保留,而那些不必要的连接权将逐渐衰减为零。最后可去掉那些影响不大的连接权和相应的节点,从而得到一个适合规模的网络结构。

若采用单隐含层的 BP 网络,使得隐含层的节点数目太大时,可应用两层隐含层的 BP 网络。一般而言,采用两层隐含层的节点总数比采用一层隐含层所用的节点数少。

网络的节点数对网络的泛化能力影响很大,节点数太多,它倾向于记住所有的训练数据,包括噪声的影响,反而降低了泛化能力;而节点数太少,它不能拟合样本数据,因此也谈不上有较好的泛化能力。

2. 确定网络的初始权值 ω_{ij}

BP 网络的各层初始权值一般选取一组较小的非零随机数。为了避免出现局部极值问题,可选取多组初始权值,最后选用最好的一种。

3. 产生训练样本

一个性能良好的神经网络离不开学习,神经网络的学习是针对样本数据进行学习的。因此,数据样本对于神经网络的性能有着至关重要的影响。

建立样本数据之前,首先要收集大量的原始数据,并在大量的原始数据中确定出最主要的输入模式,分析数据的相关性,选择其中最主要的输入模式,并确保所选择的输入模式互不相同。

在确定了最重要的输入模式后,需要进行尺度变换和预处理。在进行尺度变换之前,必须检查是否存在异常点。如果存在异常点,则异常点必须剔除。通过对数据的预处理分析还可以检验所选择的输入模式是否存在周期性、固定变化趋势或其他关系。对数据的预处理就是要对数据进行变换,从而使神经网络更容易学习和训练。

对于一个复杂问题,应该选择多少个数据,也是一个关键性问题。系统的输入/输出关系就包含在样本数据中。所以一般来说,取的数据越多,学习和训练的结果越能正确反映输入/输出关系。但是选太多的数据将增加收集、分析数据及网络训练所付出的代价。当然,选择太少的数据则可能得不到正确的结果。事实上数据的多少取决于许多因素,如网络的大小、网络测试的需要和输入/输出的分布等。其中,网络的大小是最关键的因素。通常较大的网络需要较多的训练数据。经验规则:训练模式应是连接权总数的 3～5 倍。

样本数据包含两部分:一部分用于网络的训练;另一部分用于网络的测试。测试数据应是独立的数据集合。一般而言,将收集到的样本数据随机地分成两部分,一部分作训练数据,则另一部分可作为测试数据。

影响样本数据大小的另一个因素是输入模式和输出结果的分布,对数据预先加以分类可以减少所需的数据量。相反,数据稀薄不匀甚至互相覆盖,则势必要增加数据量。

4. 训练网络

在对网络进行训练的过程中，训练样本需要反复使用。对所有训练样本数据正向运行一次并反传修改连接权一次称为一次训练（或一次学习），这样的训练需要反复进行，直至获得合适的映射结果。通常训练一个网络需要多次。

特别应该注意的是，并非训练的次数越多，越能得到正确的输入/输出的映射关系。训练网络的目的在于找出蕴含在样本数据中的输入和输出之间的本质联系，从而对于未经训练的输入也能给出合适的输出，即具备泛化功能。由于所收集的数据都是包含噪声的，训练的次数过多，网络会将包含噪声的数据都记录下来，在极端的情况下，训练后的网络可以实现查表的功能。但是，对于新的输入数据却不能给出合适的输出，即并不具有很好的泛化能力。网络的性能主要用它的泛化能力来衡量，并不是用对训练数据的拟合程度来衡量，而是要用一组独立的数据来加以测试和检验。

5. 测试网络

用一组独立的测试数据测试网络的性能，在测试时需要保持连接权系数不改变，只用该数据作为网络的输入，正向运行该网络，检验输出的均方误差。

6. 判断网络

在实际确定 BP 网络时，通常应将训练和测试交替进行，即每训练一次，同时用测试数据测试一遍网络，画出均方误差随训练次数的变化曲线，如图 6-19 所示。从误差曲线来看，在用测试数据检验时，均方误差开始逐渐减小，当训练次数再增加时，测试检验误差反而增加。误差曲线上极小点所对应的即为恰当的训练次数，若再训练即为“过度训练”。

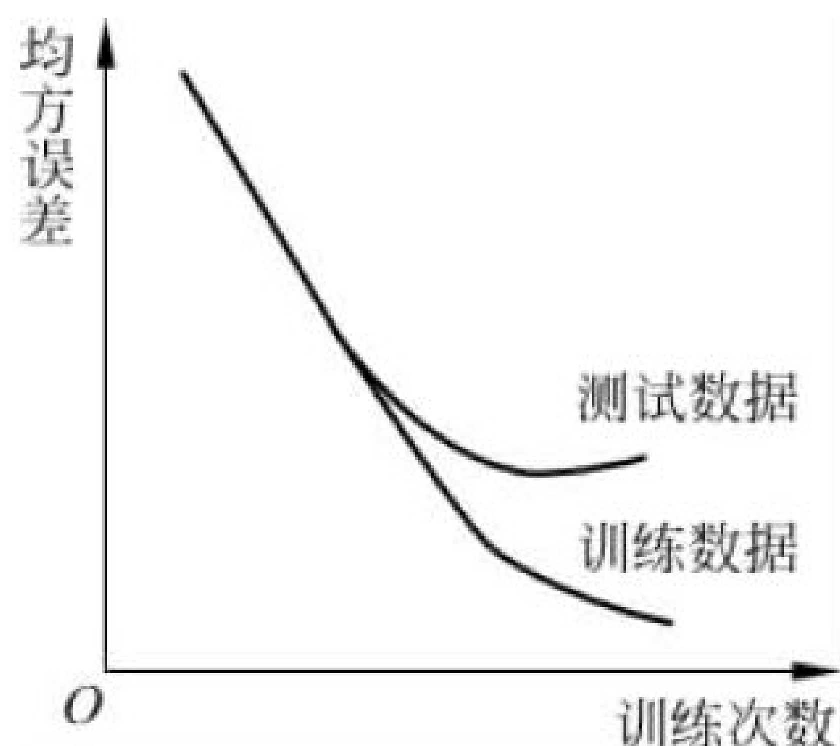


图 6-19 均方误差曲线

6.3.4 BP 网络分类器的 MATLAB 实现

在人工神经网络的实际应用中，BP 网络广泛应用于函数逼近、模式识别/分类、数据压缩等。80%～90%的人工神经网络模型采用 BP 网络或它的变化形式，它也是前馈网络的核心部分，体现了人工神经网络最精华的部分。

下面使用 MATLAB 构建 BP 神经网络。

1. 网络的构建

首先需要构造一个网络构架，函数 newff()就是用于构建神经网的。

它需要四个输入条件，依次是：由 R 维的输入样本最大最小值构成的 $R\times 2$ 维矩阵、各层的神经元个数、各层神经元的传递函数以及训练用函数的名称。下面具体介绍这些参数及其选择。

网络层数：BP 网络可以包含不同的隐层。但理论上已经证明，在不限制隐层节点数的情况下，两层（只有一个隐层）的 BP 网络可以实现任意非线性映射。因此选用两层 BP 网络

即可。

输入层节点数 m：在输入层起缓冲存储器的作用，它接受外部的输入数据，因此其节点数取决于矢量的维数。

输出层节点数 n：输出层的节点数取决于两个方面，输出数据类型和表示该类型所需数据大小。当 BP 网络用于模式分类时，则输出层的节点数可根据待分类模式数来确定。

隐含层节点数：一般认为，隐层节点数与求解问题的要求、输入输出单元数多少都有直接的关系。对于用于模式识别/分类的 BP 网络，根据前人经验，可以参照公式 $S_1=\sqrt{n+m}+a$ 进行设计。其中，m 为输入层节点数；n 为输出层节点数；a 为 1～10 的常数。

传输函数：BP 网络中的传输函数通常采用 S 形函数 $f(x)=\frac{1}{1+e^{-x}}$，在某些特定情况下还可能采用纯线性(pureline)函数。

训练函数：BP 神经网络的训练函数有 traingd、traingdm、traingdx、trainrp、traincgf、traincgp、traincgb、trainscg、trainbfg、trainoss、trainlm、trainbr 等，每种训练函数各有特点，但是没有一种函数能适应所有情况下的训练过程。代码如下：

```
net = newff(minmax(p),[12,4],{'tansig','logsig'},'trainlm');
```

2. 网络的初始化

网络的输入向量：$\boldsymbol{P}_k=(a_1,a_2,\cdots,a_n)$。

网络的目标向量：$\boldsymbol{t}_k=(y_1,y_2,\cdots,y_q)$。

网络初始化程序：net=init(net)。

将所用的数据以文本文件的形式输入，如果所收集的数据不在同一数量级，要进行归一化处理。归一化是为了加快训练网络的收敛性，也可以不进行归一化处理。归一化的具体作用是归纳统一样本的统计分布性。归一化在 0～1 是统计的概率分布，归一化在 −1～+1 是统计的坐标分布。归一化有同一、统一和合一的意思。无论是为了建模还是为了计算，首先基本度量单位要统一，神经网络是以样本在事件中的统计分布概率来进行训练(概率计算)和预测的，归一化是统一在 0～1 的统计概率分布。

当所有样本的输入信号都为正值时，与第一隐含层神经元相连的权值只能同时增加或减小，从而导致学习速度很慢。为了避免出现这种情况，加快网络学习速度，可以对输入信号进行归一化，使得所有样本的输入信号其均值接近于 0 或与其均方差相比很小。

归一化是因为 sigmoid 函数的取值是 0～1，网络最后一个节点的输出也是如此，所以经常要对样本的输出进行归一化处理。所以这样做分类的问题时用[0.9 0.1 0.1]就要比用[1 0 0]要好。

程序代码如下：

```
【pn,minp,maxp】= premnmx(p);（归一化处理,归一化后的数据将分布在[-1,1]区间内.)
【r,q】= size(p); % 训练输入样本集 p 的行数 r 和列数 q
【s2,q】size(t); % 训练目标样本集 t 的行数 s2 和列数 q
```

3. 训练参数初始化

隐含层节点数的确定使用公式 $S_1=\sqrt{n+m}+a$。其中,m 为输入层节点数;n 为输出层节点数;a 为 1～10 的常数。因为此处是 3 输入 4 输出的神经网络,所以隐含层节点数选择 9。代码如下:

```
max_epoch = x;           % 最大训练次数 x
err_goal = E;            % 期望误差
```

4. 网络训练

网络训练代码如下:

```
net = train(net,p,t);
```

5. 网络仿真

网络仿真代码如下:

```
y = sim(net,p_test);
```

6. 结果对比

在本例中采用表 1-2 的三元色数据,希望按照颜色数据所表征的特点,将数据按照各自所属的类别进行归类。其中,前 29 组数据已确定类别,后 30 组数据待确定类别。

在此使用 BP 网络对数据分类。BP 网络的输入和输出层的神经元数目由输入和输出向量的维数确定。输入向量由 A、B、C 这三列决定,所以输入层的神经元数目为 3;输出结果有 4 种模式,用 1、2、3、4 代表 4 种输出,因此输出层的神经元个数为 4。模式识别程序如下:

```
 % 构建训练样本中的输入向量 p
p = [1739.94  373.3  1756.77  864.45  222.85  877.88  1803.58  2352.12  401.3  363.34
    1571.17  104.8  499.85
    2297.28  2092.62  1418.79  1845.59  2205.36  2949.16  1692.62  1680.67  2802.88
    172.78  2063.54  1449.58
    1651.52  341.59  291.02  237.63;
    1675.15  3087.05  1652  1647.31  3059.54  2031.66  1583.12  2557.04  3259.94
    3477.95  1731.04  3389.83
    3305.75  3340.14  3177.21  1775.89  1918.81  3243.74  3244.44  1867.5  1575.78
    3017.11  3084.49  3199.76
    1641.58  1713.28  3076.62  3095.68  3077.78;
    2395.96  2429.47  1514.98  2665.9  2002.33  3071.18  2163.05  1411.53  2150.98
    2462.86  1735.33  2421.83
```

```
2196.22  535.62  584.32  2772.9  2226.49  1202.69  662.42  2108.97  1725.1  1984.98
2328.65  1257.21
3405.12  1570.38  2438.63  2088.95  2251.96];
%构建训练样本中的目标向量t
t=[0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1;
   1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0;
   0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 0;
   0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0];
%创建一个BP网络,隐含层有12个神经元,传递函数为tansig
%中间层有4个神经元,传递函数为logsig,训练函数为trainlm
net=newff(minmax(p),[12,4],{'tansig','logsig'},'trainlm');
%训练次数,默认为100
net.trainParam.epochs=500;
%训练的目标,默认为0
net.trainParam.goal=0.01;
%神经网络训练
 net=train(net,p,t);
%测试样本进行分类
   p_test=[1702.8  1877.93  867.81  1831.49  460.69  2374.98  2271.89  1783.64
           198.83  1494.63  1597.03  1598.93  1243.13  2336.31  354  2144.47
           426.31  1507.13  343.07  2201.94  2232.43  1580.1  1962.4  1495.18
           1125.17  24.22  1269.07  1802.07  1817.36  1860.45;

           1639.79  1860.96  2334.68  1713.11  3274.77  3346.98  3482.97  1597.99
           3250.45  2072.59  1921.52  1921.08  1814.07  2640.26  3300.12  2501.62
           3105.29  1556.89  3271.72  3196.22  3077.87  1752.07  1594.97  1957.44
           1594.39  3447.31  1910.72  1725.81  1927.4  1782.88;

           2068.74  1975.3  2535.1  1604.68  2172.99  975.31  946.7  2261.31
           2445.08  2550.51  2126.76  1623.33  3441.07  1599.63  2373.61
           591.51  2057.8  1954.51  2036.94  935.53  1298.87  2463.04  1835.95
           3498.02  2937.73  2145.01  2701.97  1966.35  2328.79  1875.83];
 y=sim(net,p_test);
```

运行上述程序代码后,可以得到网络的训练结果如下:

```
TRAINLM-calcjx,Epoch 0/500,MSE 0.303441/0.01,Gradient 173.123/1e-010
TRAINLM-calcjx,Epoch 25/500,MSE 0.0862919/0.01,Gradient 0.0209707/1e-010
TRAINLM-calcjx,Epoch460/500,MSE 0.00159/0.01,Gradient 0.226/1e-07
TRAINLM,Performance goal met.
```

图6-20所示为神经网络训练模块,在这里可以查看训练结果、训练状态等,可见到网络经过460次训练后即可达到需要的误差要求,结果如图6-21所示。从图6-21中可以看出网络具有非常好的学习性能,网络输出与目标输出的误差已经达到了需要的要求。

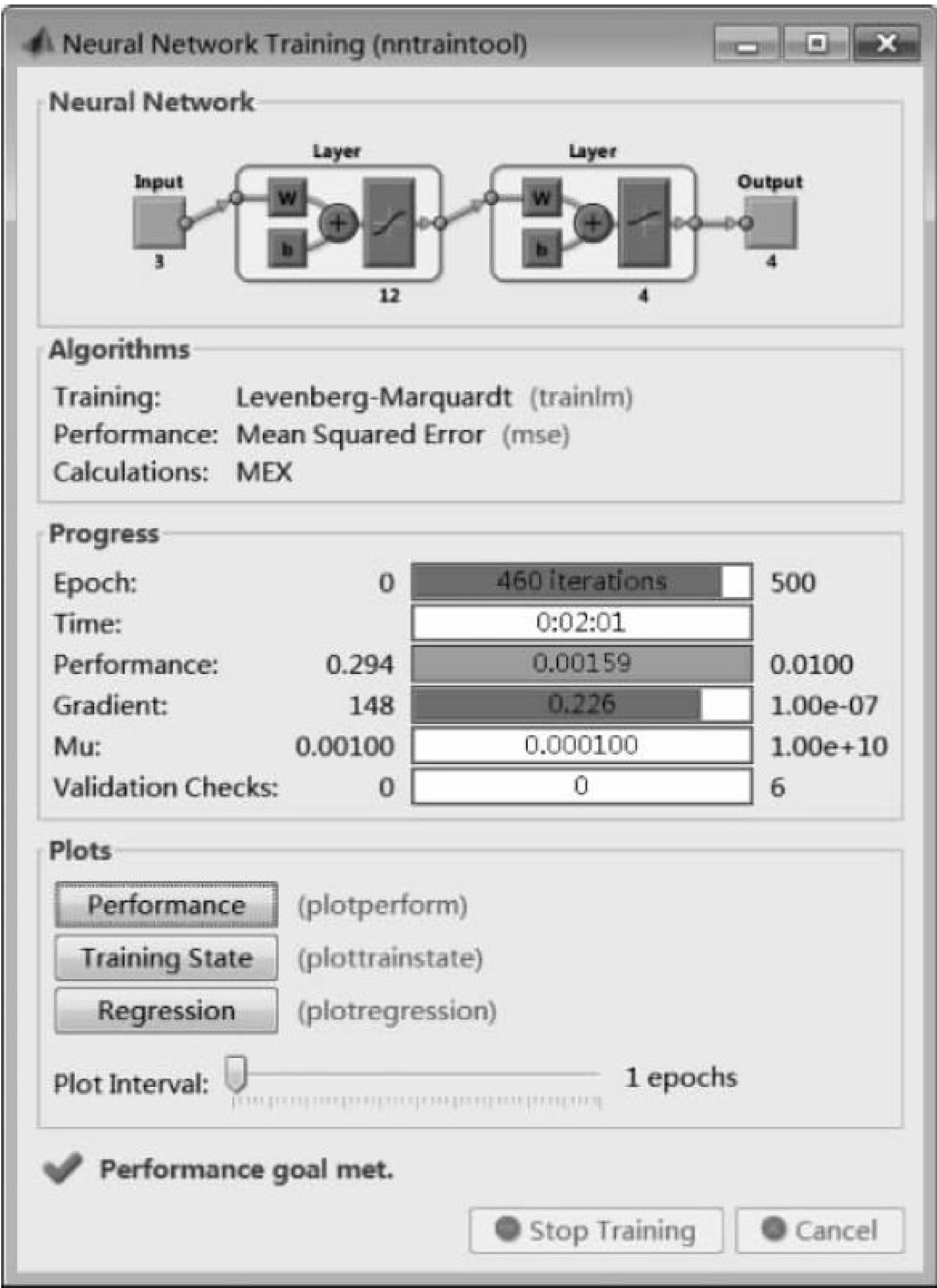


图 6-20　神经网络训练图

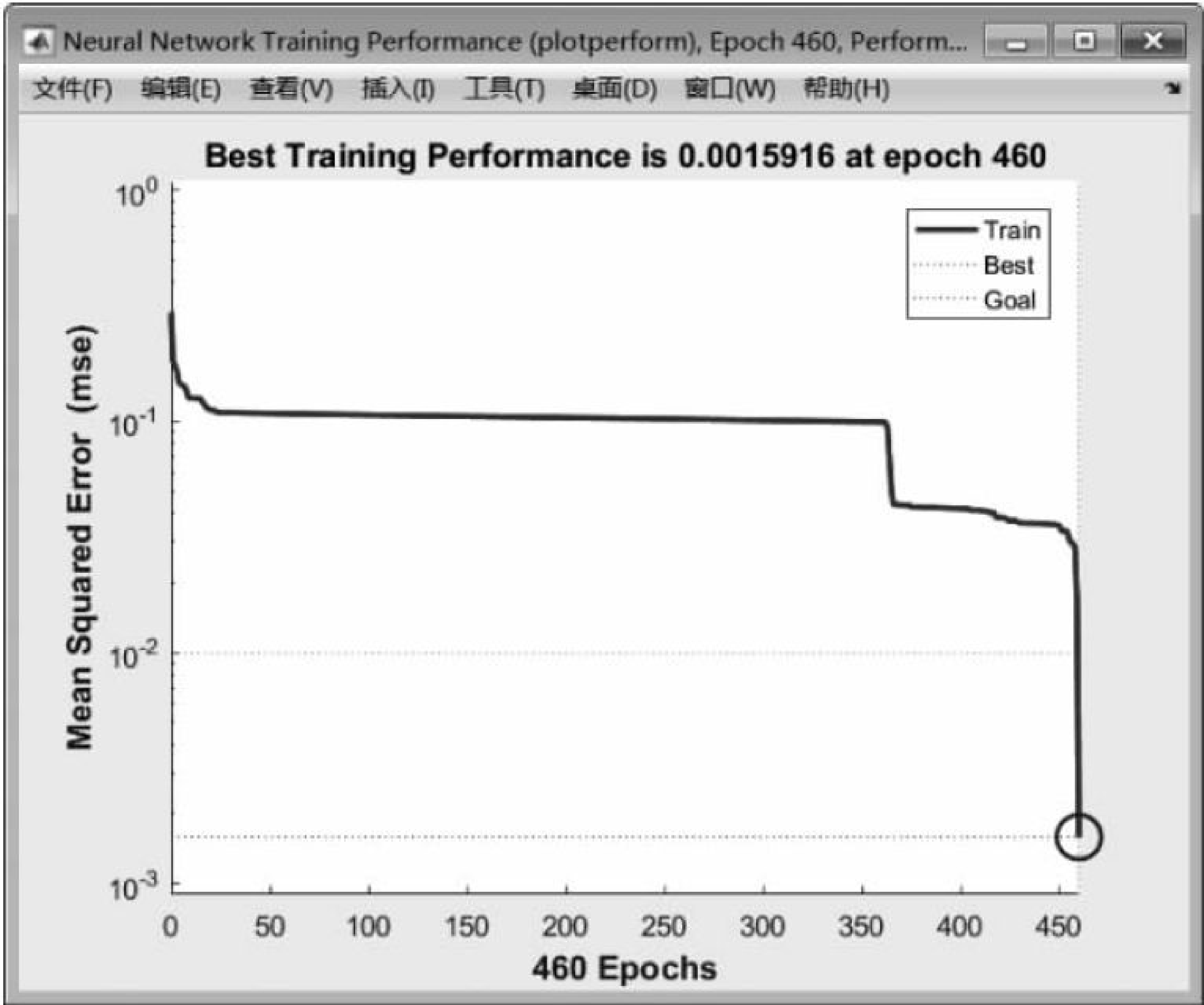


图 6-21　训练曲线图

对预测样本值的仿真输出结果如下：

```
y =

  1 ～ 18 列

    0.0144    0.0144    0.4828    0.0144    0.9788    0.0353    0.0353    0.0144
    0.9788    0.0877    0.0144    0.0239    0.0525    0.0353    0.9788    0.0353
    0.9788    0.0144    0.9857    0.9857    0.3637    0.9857    0.0187    0.0300
    0.0300    0.9857    0.0187    0.9528    0.9857    0.8961    0.1056    0.0300
    0.0187    0.0300    0.0187    0.9857    0.0057    0.0057    0.0020    0.0057
    0.0221    0.9693    0.9693    0.0057    0.0221    0.0006    0.0057    0.0822
    0.0001    0.9693    0.0221    0.9693    0.0221    0.0057    0.0264    0.0264
    0.0233    0.0264    0.0565    0.0050    0.0050    0.0264    0.0565    0.0453
    0.0264    0.0099    0.8830    0.0050    0 .0565   0.0050    0.0565    0.0264

  19 ～ 30 列

    0.9787    0.0353    0.0353    0.0144    0.0144    0.0525    0.0525    0.9788
    0.0568    0.0144    0.0144    0.0144    0.0187    0.0300    0.0300    0.9857
    0.9857    0.1056    0.1056    0.0187    0.1208    0.9857    0.9857    0.9857
    0.0221    0.9693    0.9693    0.0057    0.0057    0.0001    0.0001    0.0221
    0.0001    0.0057    0.0057    0.0057    0.0565    0.0050    0.0050    0.0264
    0.0264    0.8830    0.8830    0.0565    0.8643    0.0264    0.0264    0.0264
```

将 BP 网络的识别结果与模糊模式识别器的分类结果进行对比，结果如表 6-1 所示。

表 6-1 模糊系统分类结果与 BP 神经网络分类结果对比

序 号	A	B	C	模糊分类系统测试结果	BP 神经网络分类结果
1	1702.8	1639.79	2068.74	3	3
2	1877.93	1860.96	1975.3	3	3
3	867.81	2334.68	2535.1	1	4
4	1831.49	1713.11	1604.68	3	3
5	460.69	3274.77	2172.99	4	4
6	2374.98	3346.98	975.31	2	4/2
7	2271.89	3482.97	946.7	2	4/2
8	1783.64	1597.99	2261.31	3	1/3
9	198.83	3250.45	2445.08	4	4
10	1494.63	2072.59	2550.51	1	3
11	1597.03	1921.52	2126.76	3	3
12	1598.93	1921.08	1623.33	3	3
13	1243.13	1814.07	3441.07	1	1
14	2336.31	2640.26	1599.63	2.5	—
15	354	3300.12	2373.61	4	4
16	2144.47	2501.62	591.51	2	2
17	426.31	3105.29	2057.8	4	4
18	1507.13	1556.89	1954.51	3	3
19	343.07	3271.72	2036.94	4	4
20	2201.94	3196.22	935.53	2	2/4
21	2232.43	3077.87	1298.87	2	2/4

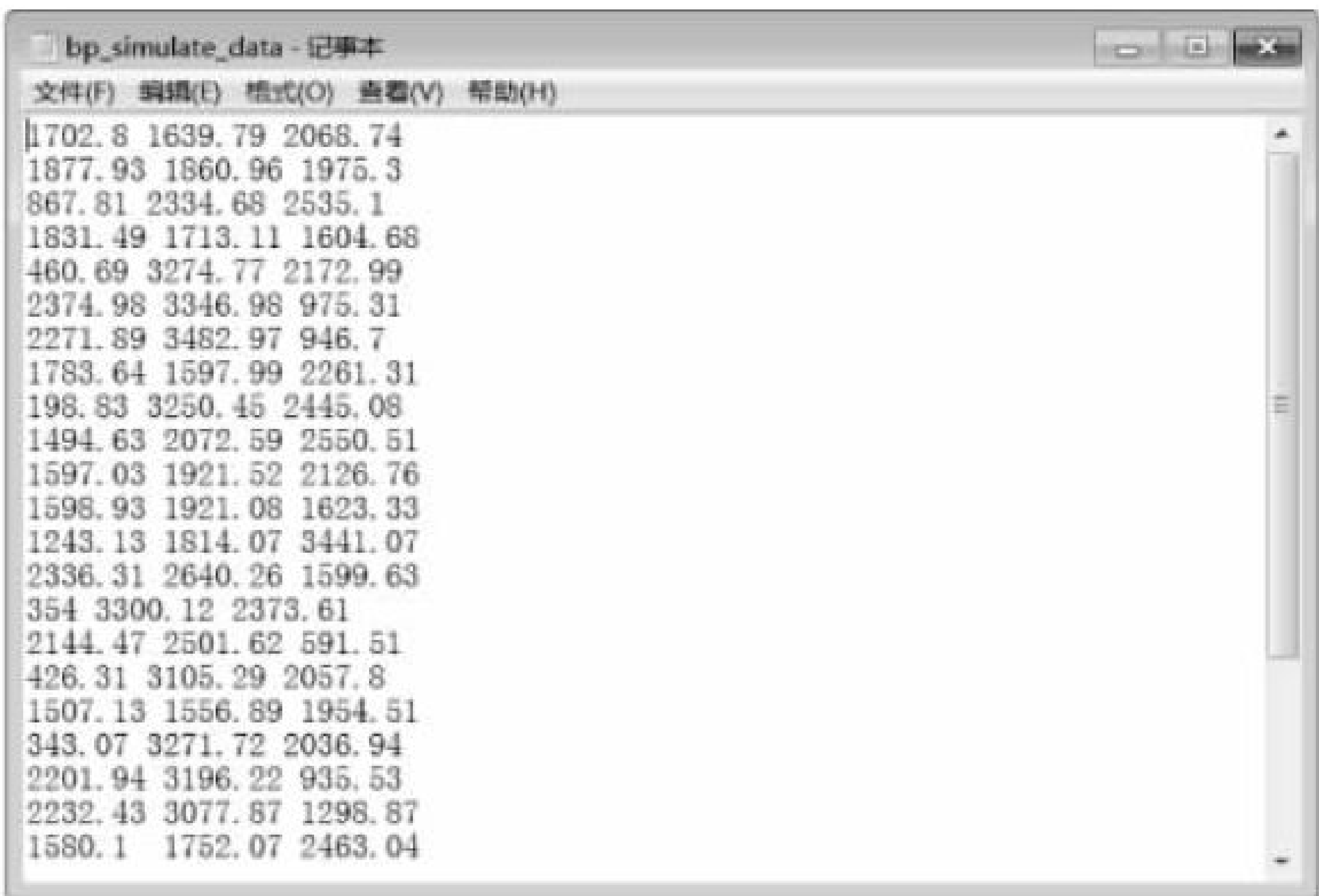
续表

序　号	A	B	C	模糊分类系统测试结果	BP 神经网络分类结果
22	1580.1	1752.07	2463.04	3	3
23	1962.4	1594.97	1835.95	3	1/3
24	1495.18	1957.44	3498.02	1	1
25	1125.17	1594.39	2937.73	1	1
26	24.22	3447.31	2145.01	4	4
27	1269.07	1910.72	2701.97	1	3
28	1802.07	1725.81	1966.35	3	3
29	1817.36	1927.4	2328.79	3	3
30	1860.45	1782.88	1875.13	3	3

从表 6-1 中的数据可以看出，系统不够理想。此时，可增加训练数据和训练次数来提高系统的识别能力。此外，还可以使用其他的训练函数训练 BP 神经网络。

6.3.5　BP 网络的其他学习算法的应用

在应用其他学习方法训练 BP 网络之前，先将样本数据(bp_train_sample_data.dat)、目标数据(bp_train_target_data.dat)及待分类数据(bp_simulate_data.dat)存放到数据文件。各文件内容及格式如图 6-22 所示。



bp_simulate_data - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
1702.8 1639.79 2068.74
1877.93 1860.96 1975.3
867.81 2334.68 2535.1
1831.49 1713.11 1604.68
460.69 3274.77 2172.99
2374.98 3346.98 975.31
2271.89 3482.97 946.7
1783.64 1597.99 2261.31
198.83 3250.45 2445.08
1494.63 2072.59 2550.51
1597.03 1921.52 2126.76
1598.93 1921.08 1623.33
1243.13 1814.07 3441.07
2336.31 2640.26 1599.63
354 3300.12 2373.61
2144.47 2501.62 591.51
426.31 3105.29 2057.8
1507.13 1556.89 1954.51
343.07 3271.72 2036.94
2201.94 3196.22 935.53
2232.43 3077.87 1298.87
1580.1  1752.07 2463.04
```

(a) bp_simulate_data.dat文件内容及格式

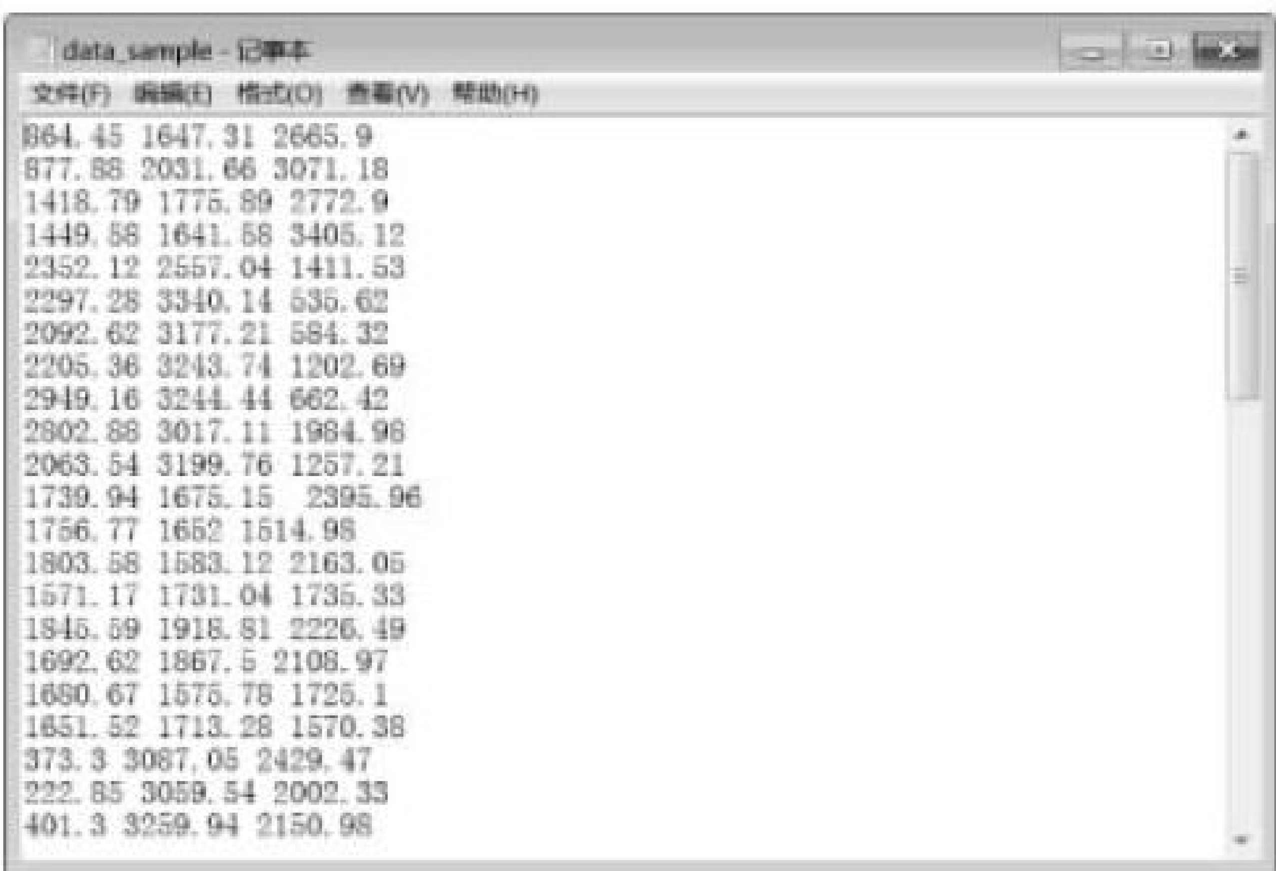


bp_train_target_data - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
1
1
1
1
2
2
2
2
2
2
2
3
3
3
3
3
3
3
3
4
4
4
```

(b) bp_train_target_data.dat文件内容及格式



data_sample - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
864.45 1647.31 2665.9
877.88 2031.66 3071.18
1418.79 1775.89 2772.9
1449.58 1641.58 3405.12
2352.12 2557.04 1411.53
2297.28 3340.14 535.62
2092.62 3177.21 584.32
2205.36 3243.74 1202.69
2949.16 3244.44 662.42
2802.88 3017.11 1984.98
2063.54 3199.76 1257.21
1739.94 1675.15  2395.96
1756.77 1652 1514.98
1803.58 1583.12 2163.05
1571.17 1731.04 1735.33
1845.59 1918.81 2226.49
1692.62 1867.5 2108.97
1680.67 1575.78 1725.1
1651.52 1713.28 1570.38
373.3 3087.05 2429.47
222.85 3059.54 2002.33
401.3 3259.94 2150.98
```

(c) data_sample.dat文件内容及格式

图 6-22　数据文件内容及格式

1. 采用梯度法进行学习

前向神经网络 BP 算法采用最速下降寻优算法，即梯度法。假设有 N 对学习样本，采取批处理学习方法，目标函数为 $E=\frac{1}{2N}\sum_{K=1}^{N}(\boldsymbol{T}_K-\boldsymbol{Y}_K)^2$，其中 $\boldsymbol{T}_K$、$\boldsymbol{Y}_K$ 分别为第 K 对样本的期望输出和实际输出向量。E 反映网络输出与样本的总体误差。学习过程就是通过修改各神经元之间的权值，使得目标函数 E 的值最小，权值按下列公式修正

$$\Delta\omega_{ij}=-\eta\frac{\partial E}{\partial\omega_{ij}} \tag{6-27}$$

其中，η 为学习速率。

应用 traingd 函数训练，应调整全值和阈值沿着表现函数的负梯度方向。如果应用梯度下降法训练函数，需要在训练之前将网络构成函数的相应参数 trainFcn 设置为 traingd。

与函数 traingd 有关的训练参数有：epochs、goal、lr、max_fail、min_grad、show、time，如果不设置就表示应用默认值。

net. trainParam. epochs	最大训练次数(默认为 10)
net. trainParam. goal	训练要求精度(默认为 0)
net. trainParam. lr	学习率(默认为 0.01)
net. trainParam. max_fail	最大失败次数(默认为 5)
net. trainParam. min_grad	最小梯度要求(默认为 1e−10)
net. trainParam. show	显示训练迭代过程(NaN 表示不显示，默认为 25)
net. trainParam. time	最大训练时间(默认为 inf)

其中学习速率是很重要的参数，它和负梯度的乘积决定了权值和阈值的调整量，学习速率越大，调整步伐越大。学习速率过大，算法会变得不稳定；但是如果学习速率太小，算法收敛的时间就会增加。

训练过程中，只要满足下面 5 个条件之一，训练就会停止：

(1) 超过最大迭代次数 epochs。

(2) 表现函数值小于误差指标 goal。

(3) 梯度值小于要求精度 mingrad。

(4) 训练所用时间超过时间限制 time。

(5) 最大失败次数超过次数限制 max_fail。

在 MATLAB 中创建 BP 网络调用相应的函数，代码如下：

```
function f = bpfun()
 % Neural Network
 % build train and simulate
 % bpfun.m
 % 输入矩阵的范围(数据源)
P = [20 3000;1400 3500;500 3500;];
 % 创建网络
net = newff(P,[12 4 1],{'tansig' 'tansig' 'purelin','traingd'});
 % 初始化神经网络
```

```
net = init(net);
%设置训练的参数
%停止方式按键
%pause;
%两次显示之间的训练步数默认为25
net.trainParam.show = 50;
%lr不能选择太大,太大了会造成算法不收敛,太小了会使训练时间太长
%一般选择0.01～0.1
%训练速度
net.trainParam.lr = 0.05;
%训练次数,默认为100
net.trainParam.epochs = 3000;
%训练时间,默认为inf,表示训练时间不限
net.trainParam.time = 6000;
%训练的目标,默认为0
net.trainParam.goal = 0.001;
%建立源数据的矩阵
SourceDataConvert = importdata('bp_train_sample_data.dat');
SourceData = SourceDataConvert'
TargetConvert = importdata('bp_train_target_data.dat');
Target = TargetConvert'
%神经网络训练
net = train(net,SourceData,Target)
%显示训练后的各层权重
mat1 = cell2mat(net.IW(1,1))
mat2 = cell2mat(net.LW(2,1))
mat3 = cell2mat(net.LW(3,2))
%读取仿真文件数据
simulate_data_convert = importdata('bp_simulate_data.dat');
simulate_data = simulate_data_convert';
result = sim(net,simulate_data)
```

多次运行上述程序,可以得到满足误差要求的网络的训练结果:

```
T TRAINLM - calcjx,Epoch 0/3000,Time 0.0%,MSE 14.4178/0.001,Gradient 10741.9/1e-010
TRAINLM - calcjx,Epoch 40/3000,Time 0.0%,MSE 0.000438/0.001,Gradient 0.196/1e-07
TRAINLM,Performance goal met
```

图6-23展示了神经网络训练模块,在这里可以查看训练结果、训练状态等。训练后即可达到误差要求,结果如图6-24所示。

对预测样本值的仿真输出结果如下:

```
result =
1 ～ 9 列
3.0016    3.0016    0.9969    3.0011    3.9847    1.9460    1.9460    3.0016    3.9873
10 ～ 18 列
0.9973    3.0016    2.9982    0.9969    2.0245    3.9873    1.9460    3.9850    3.0016
19 ～ 27 列
3.8844    1.9460    2.0245    2.5884    3.0016    0.9969    0.9969    3.9816    0.9969
28 ～ 31 列
3.0016    3.0016    3.0016    3.0015
```

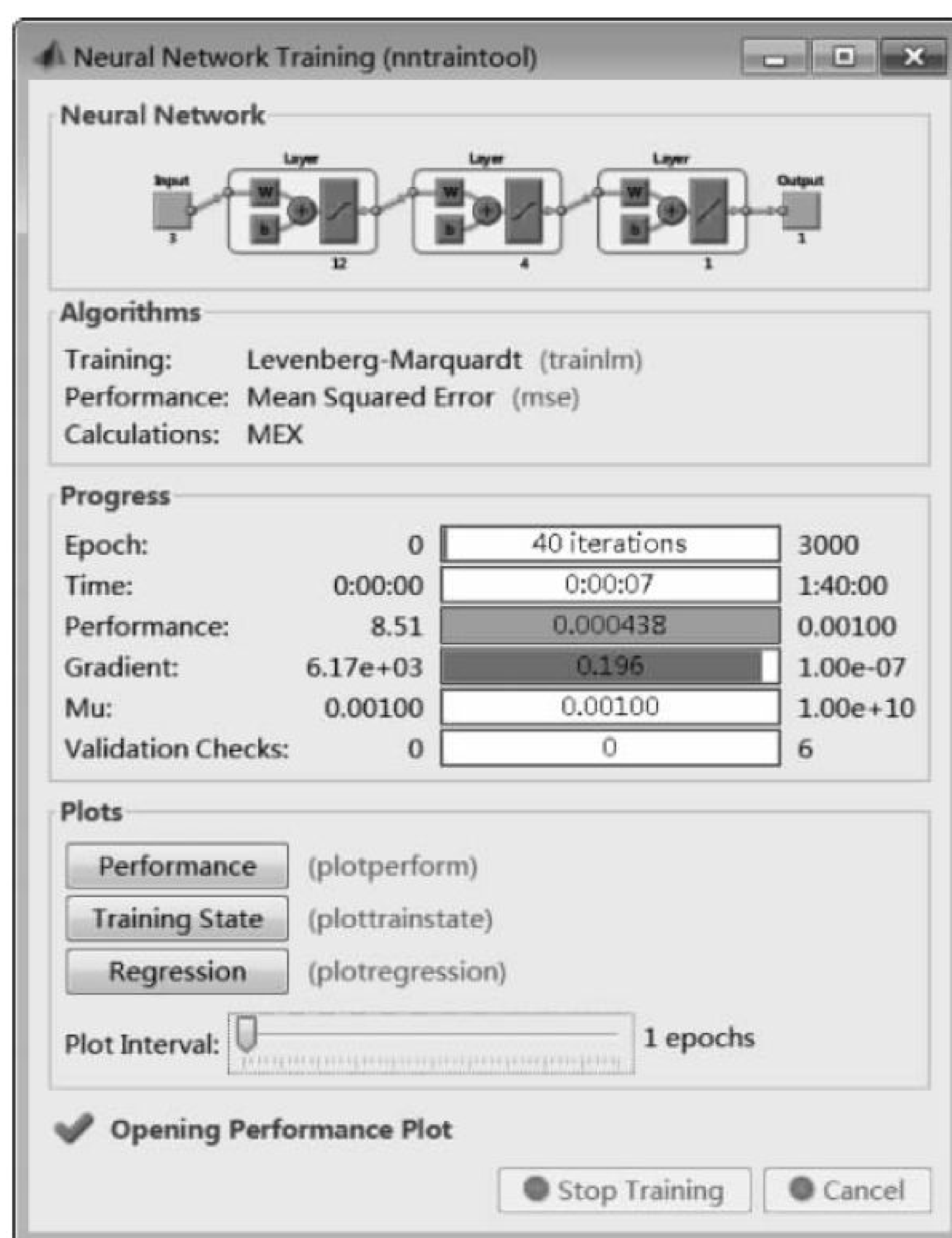


图 6-23　神经网络训练模块

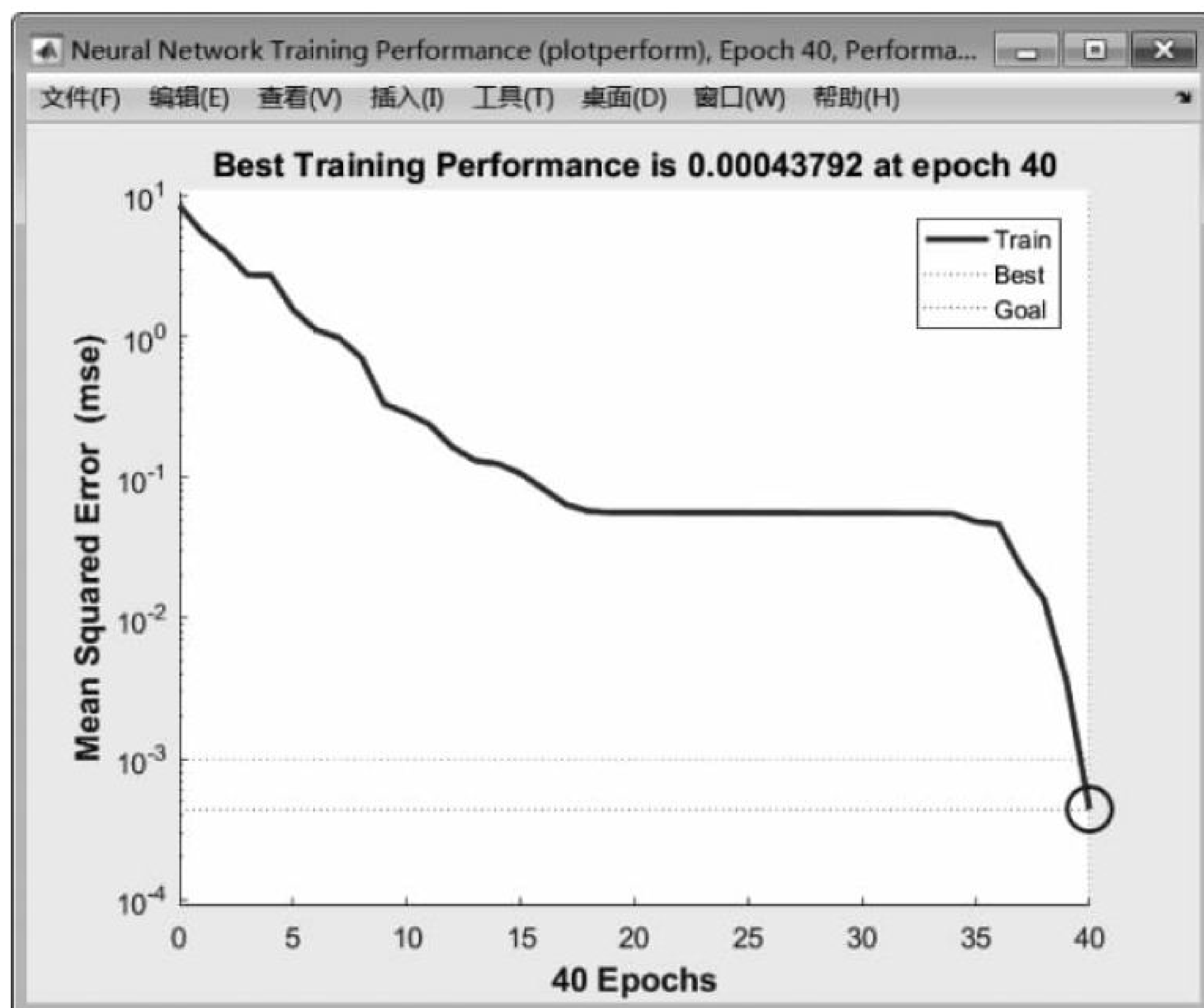


图 6-24　训练曲线图

2. 采用带动量最速下降法进行学习

带动量最速下降法在非二次型较强的区域能使目标函数收敛较快。BP 算法的最速下降方向即目标函数 E 在权值空间上的负梯度方向，在无约束优化目标函数 E 时，相邻的两个搜索方向正交。因此，当权值接近于极值区域时，每次迭代移动的步长很小，呈现出“锯齿”现象，严重影响了收敛速率，有时甚至不能收敛而在局部极值区域振荡。为此，提出了各种加速学习速率的优化算法，其中加动量项的算法为当前广为应用的方法，其权值修正公式为 $\Delta\omega_{ij}(t)=-\eta\frac{\partial E}{\partial\omega_{ij}}+\alpha\Delta\omega_{ij}(t-1)$，$\alpha$ 为动量系数。引入动量项后，使得调节向着底部的平均方向变化，不致产生大的摆动，即起到缓冲平滑的作用。若系统进入误差函数面的平坦区，那么误差变化将很小，动量项的引入使得调节尽快脱离这一平坦区，有助于缩短向极值逼近的时间。所以，动量项的引入，加快了学习速度。

在训练过程中，若能选择合适的速率，使它的值尽可能大但又不至于引起振荡，则能使训练快速达到要求。

在 MATLAB 中创建 BP 网络的程序代码如下：

```
function f = bpfun()
%Neural Network
%build train and simulate
%bpfun.m
%输入矩阵的范围(数据源)
P = [20 3000;1400 3500;500 3500;];
%创建网络
net = newff(P,[12 4 1],{'tansig' 'tansig' 'purelin','traingdx'});
%初始化神经网络
net = init(net);
%设置训练的参数
%停止方式按键
%pause;
%两次显示之间的训练步数默认为 25
net.trainParam.show = 50;
%lr 不能选择太大,太大了会造成算法不收敛,太小了会使训练时间太长
%一般选择 0.01～0.1
%训练速度
net.trainParam.lr = 0.05;
%速度增长系数
net.trainParam.lr_inc = 1.2;
%速度下调系数
net.trainParam.lr_dec = 0.8;
%添加动量因子
net.trainParam.mc = 0.9;
%训练次数,默认为 100
net.trainParam.epochs = 3000;
%训练时间,默认为 inf,表示训练时间不限
net.trainParam.time = 6000;
%训练的目标,默认为 0
```

```
net.trainParam.goal = 0.001;
%建立源数据的矩阵
SourceDataConvert = importdata('bp_train_sample_data.dat');
SourceData = SourceDataConvert'
TargetConvert = importdata('bp_train_target_data.dat');
Target = TargetConvert'
%神经网络训练
net = train(net,SourceData,Target)
%显示训练后的各层权重
mat1 = cell2mat(net.IW(1,1))
mat2 = cell2mat(net.LW(2,1))
mat3 = cell2mat(net.LW(3,2))
%读取仿真文件数据
simulate_data_convert = importdata('bp_simulate_data.dat');
simulate_data = simulate_data_convert';
result = sim(net,simulate_data)
```

多次运行上述程序,可以得到满足误差要求的网络的训练结果:

```
TRAINLM - calcjx, Epoch 0/3000, Time 0.0%, MSE 14.0262/0.001, Gradient 7315.37/1e-010
TRAINLM - calcjx, Epoch 335/3000, Time 0.0%, MSE 0.000442/0.001, Gradient1.19/1e-7
TRAINLM, Performance goal met
```

神经网络训练工具如图 6-25 所示,训练后即可达到误差要求,结果如图 6-26 所示。

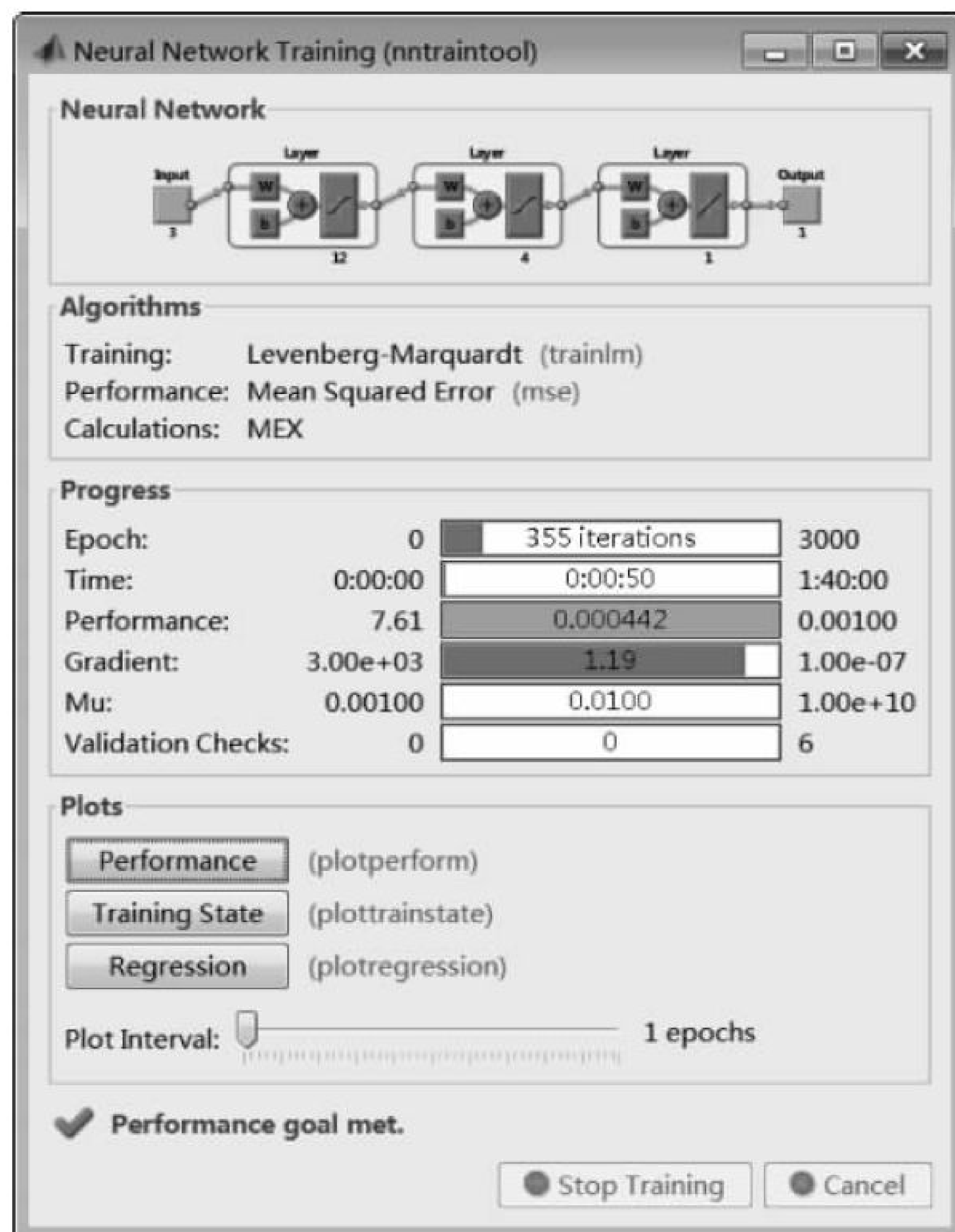


图 6-25 神经网络训练工具箱

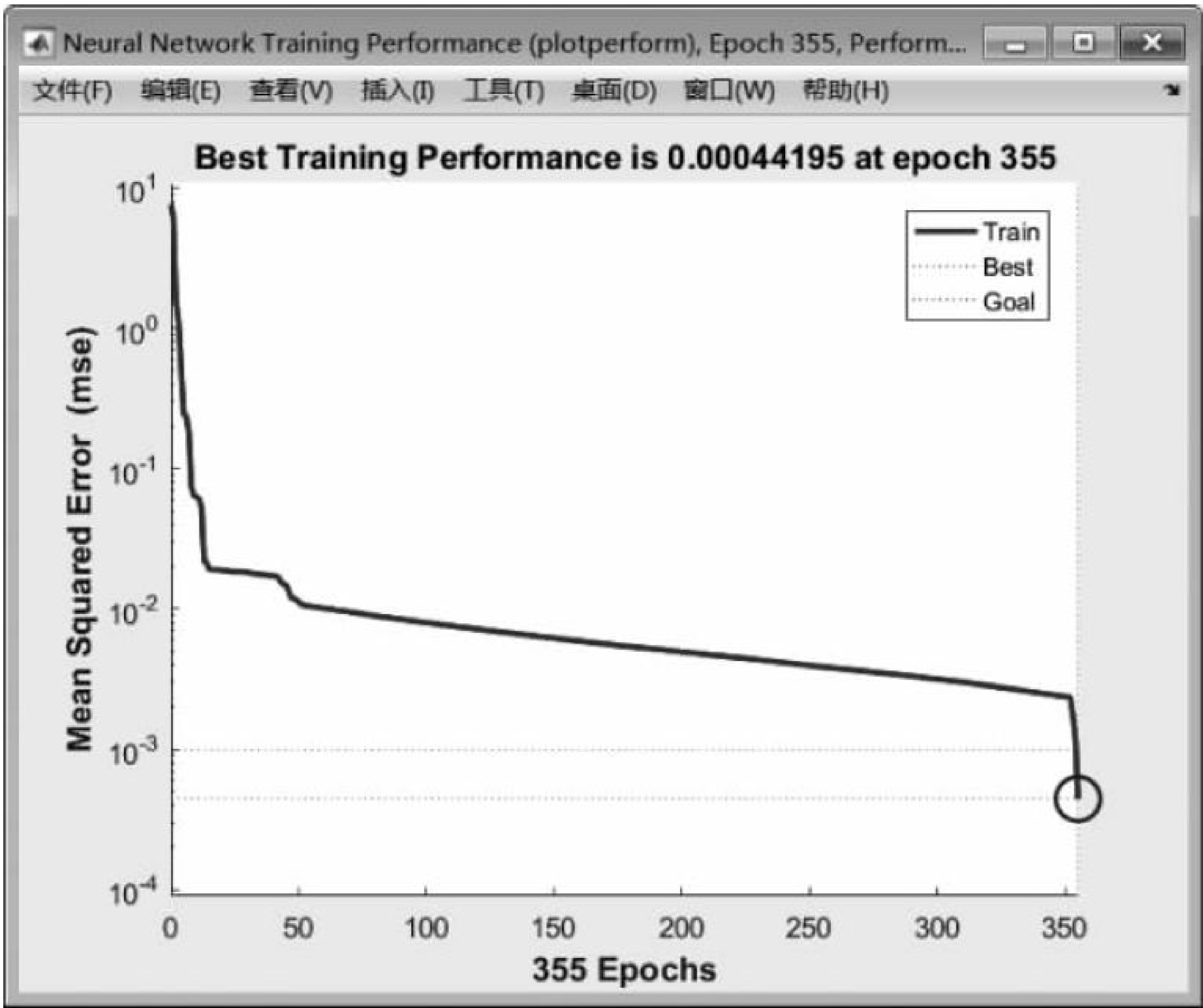


图 6-26 训练曲线图

对预测样本值的仿真输出结果如下：

```
result =
1 ～ 9 列
3.0296    2.9309    1.5220    2.9256    3.9876    1.9889    1.9904    3.0315    3.9897
10 ～ 18 列
1.3982    3.0274    2.5501    1.0009    2.0562    3.9843    2.0376    3.9917    3.0733
19 ～ 27 列
3.9915    2.0045    2.0212    1.8679    2.9285    0.9716    1.0037    4.0013    1.4880
28 ～ 31 列
2.9780    2.9835    2.9379    3.0032
```

3. 采用共轭梯度法进行学习

共轭梯度法是重要的无约束优化方法，它利用一维搜索所得到的极小点处的最速下降方向生成共轭方向，并据此搜索目标函数极值。共轭梯度法的计算步骤和梯度法的计算步骤差别不大，主要差别在于搜索方向的不同，即每一步的方向不再是梯度方向，而是一种共轭的方向，由原来的负梯度方向加上一个修正项（前一点的梯度乘以适当的系数）得到共轭方向。设梯度向量为 $\boldsymbol{g}$，共轭向量为 $\boldsymbol{P}$，则第 k 次的共轭方向为

$$P_k = \begin{cases} -g_k, & k=0 \\ -g_k + \beta_{k-1}P_{k-1}, & k \geqslant 1 \end{cases} \tag{6-28}$$

其中，$\beta_{k-1} = g_k g_k^{\mathrm{T}} / g_{k-1} g_{k-1}^{\mathrm{T}}$ 为标量，其大小必须保证 P_k 和 P_{k-1} 为共轭方向。因此，可以说共轭梯度法综合利用过去的梯度和现在某点的梯度信息，用其线性组合来构造更好的搜索

方向，这样权值的修正公式就为 $\omega_{ij}(k)=\omega_{ij}(k-1)+\eta p_k$。

共轭梯度法在二次型较强的区域能使目标函数收敛较快。而一般目标函数在极小点附近的形态近似于二次函数，故共轭梯度法在极小点附近有较好的收敛性。

在 MATLAB 中创建 BP 网络的程序代码如下：

```
function f = bpfun()
 % Neural Network
 % build train and simulate
 % bpfun.m
 % 输入矩阵的范围(数据源)
P = [20 3000;1400 3500;500 3500;];
 % 创建网络
net = newff(P,[12 4 1],{'tansig' 'tansig' 'purelin','traincgb'});
 % 初始化神经网络
net = init(net);
 % 设置训练的参数
 % 停止方式按键
 % pause;
 % 两次显示之间的训练步数默认为 25
net.trainParam.show = 50;
 % lr 不能选择太大,太大了会造成算法不收敛,太小了会使训练时间太长
 % 一般选择 0.01～0.1
 % 训练速度
net.trainParam.lr = 0.05;
 % 训练次数,默认为 100
net.trainParam.epochs = 3000;
 % 训练时间,默认为 inf,表示训练时间不限
net.trainParam.time = 6000;
 % 训练的目标,默认为 0
net.trainParam.goal = 0.001;
 % 建立源数据的矩阵
SourceDataConvert = importdata('bp_train_sample_data.dat');
SourceData = SourceDataConvert'
TargetConvert = importdata('bp_train_target_data.dat');
Target = TargetConvert'
 % 神经网络训练
net = train(net,SourceData,Target)
 % 显示训练后的各层权重
mat1 = cell2mat(net.IW(1,1))
mat2 = cell2mat(net.LW(2,1))
mat3 = cell2mat(net.LW(3,2))
 % 读取仿真文件数据
simulate_data_convert = importdata('bp_simulate_data.dat');
simulate_data = simulate_data_convert';
result = sim(net,simulate_data)
```

多次运行上述程序，可以得到满足误差要求的网络训练结果：

```
TRAINLM - calcjx,Epoch 0/3000,Time 0.0 % ,MSE 14.0262/0.001,Gradient 7315.37/1e - 010
TRAINLM - calcjx,Epoch 7/3000,Time 0.0 % ,MSE 6.11e - 0.5/0.001,Gradient 0.185/1e - 07
TRAINLM,Performance goal met
```

图 6-27 展示了神经网络训练模块，在这里可以查看训练结果、训练状态等。训练后即可达到误差要求，结果如图 6-28 所示。

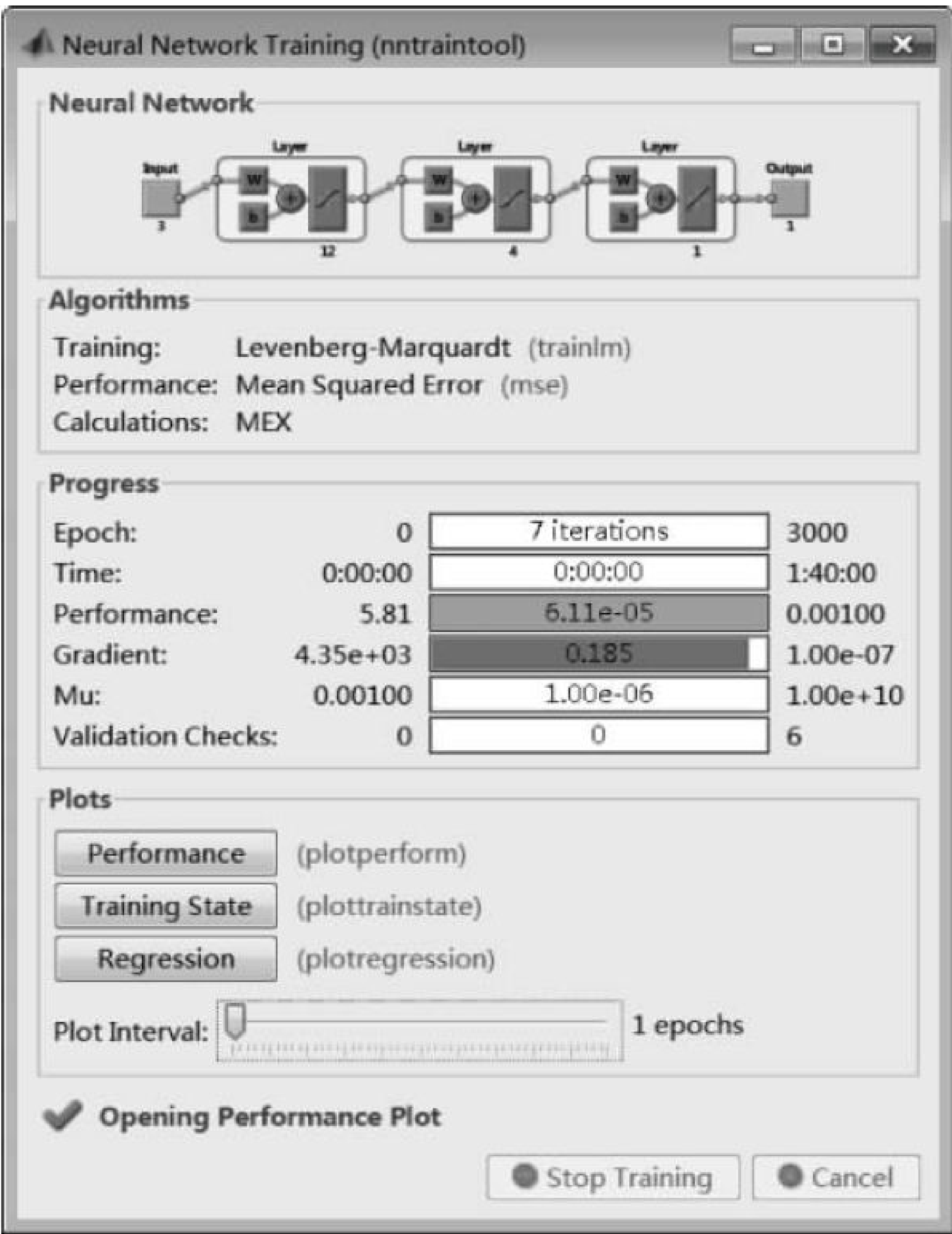


图 6-27 神经网络训练模块

对预测样本值的仿真输出结果如下：

```
result =
1 ～ 9 列
2.9953    2.9953    1.0002    2.9962    3.9875    2.6118    2.6118    2.9953    4.0590
10 ～ 18 列
3.2461    2.9953    3.4248    1.0096    1.9975    3.9875    2.0003    3.9875    2.9953
19 ～ 27 列
3.9875    2.6118    1.9975    3.0221    2.9953    1.0011    1.0137    3.9888    1.0001
28 ～ 31 列
2.9953    2.9953    2.9953    1.9339
```

综上所述，在计算过程的第一阶段，最速下降法是比较理想的寻优方法，而在最优点附近，由于接近于二次型函数，宜采用共轭梯度法。

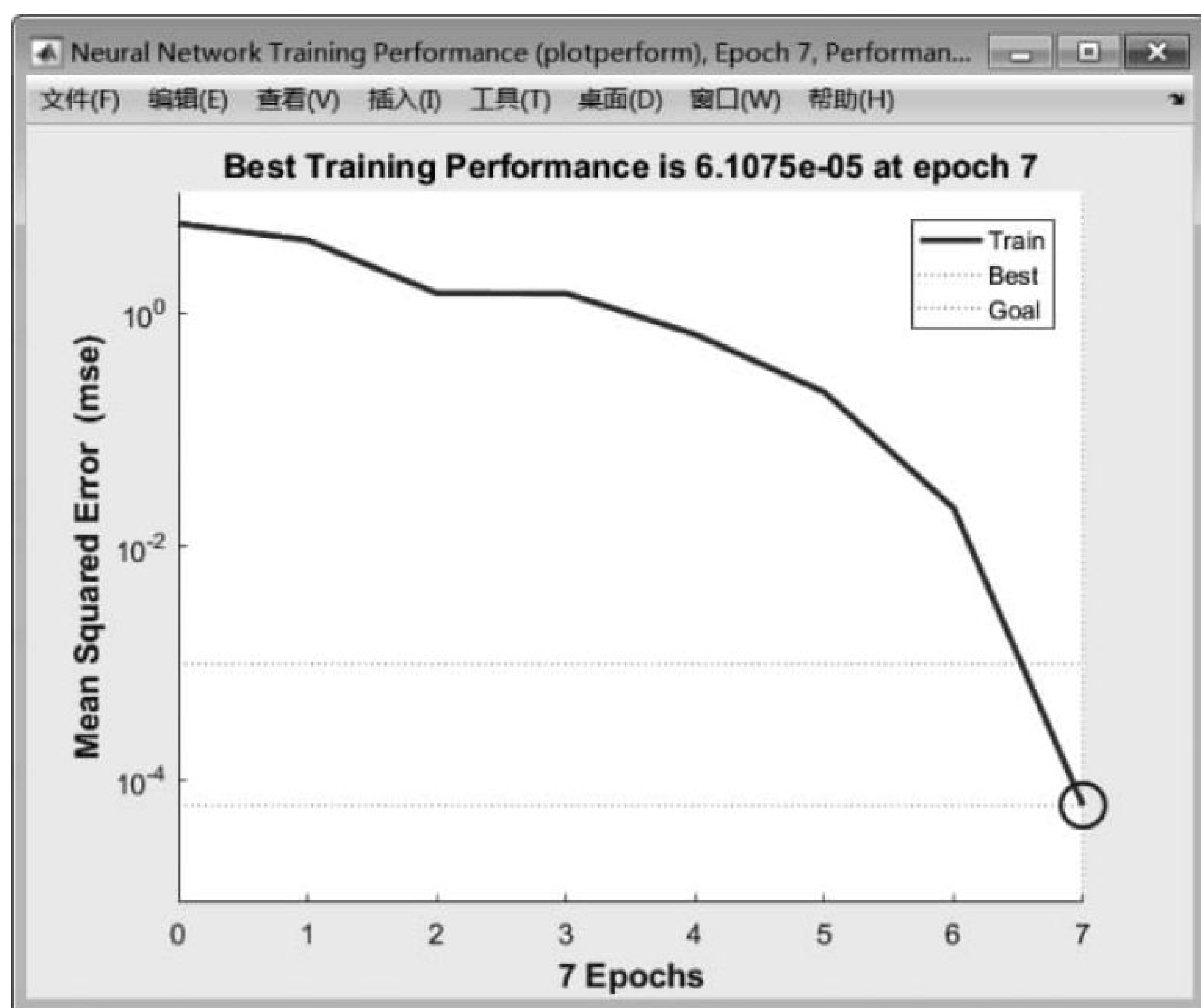


图 6-28　训练曲线图

6.4 反馈神经网络

由于反馈网络的输出端有反馈到其输入端，即该网络在输入的激励下，会产生不断的状态变化，因此反馈神经网络需要工作一段时间才能达到稳定状态。需要指出的是，反馈网络有稳定的，也有不稳定的。

反馈神经网络首先由 Hopfield 提出，因此通常称反馈网络为 Hopfield 网络。在这种网络模型的研究中，首次引入了网络能量函数的概念，并给出了网络稳定性的判据。1984 年，Hopfield 提出了网络模型实现的电子电路，为神经网络的工程实现指明了方向。这种网络是反馈网络的一种，所有神经单元之间相互连接，具有丰富的动力学特性。现在，Hopfield 网络已经广泛应用于联想记忆和优化计算中，取得了很好的效果。根据网络的输入是连续量还是离散量，Hopfield 网络也分为连续 Hopfield 网络和离散 Hopfield 网络。这里以离散 Hopfield 网络为例进行讲解。

6.4.1 离散 Hopfield 网络的结构

Hopfield 最早提出的网络是二值神经网络，神经元的输出只取 0 和 1(或－1 和 1)两个值，也称为离散 Hopfield 网络。离散 Hopfield 网络是一种单层的输入输出为二值的反馈网络，主要用于联想记忆。网络的能量函数存在着一个或多个极小点，或者称为平衡点。当网络的初始状态确定后，网络状态按其工作规则向能量递减的方向变化，最后接近或达到平衡

点。这种平衡点又称为吸引子。如果设法把网络所需记忆的模式设计成某个确定网络状态的一个平衡点，则当网络从与记忆模式较接近的某个初始状态出发后，按 Hopfield 运行规则进行状态更新，最后网络状态稳定在能量函数的极小点，即记忆模式所对应的状态。这样就完成了由部分信息或失真的信息到全部或完整信息的联想记忆过程。

离散 Hopfield 网络的结构如图 6-29 所示。

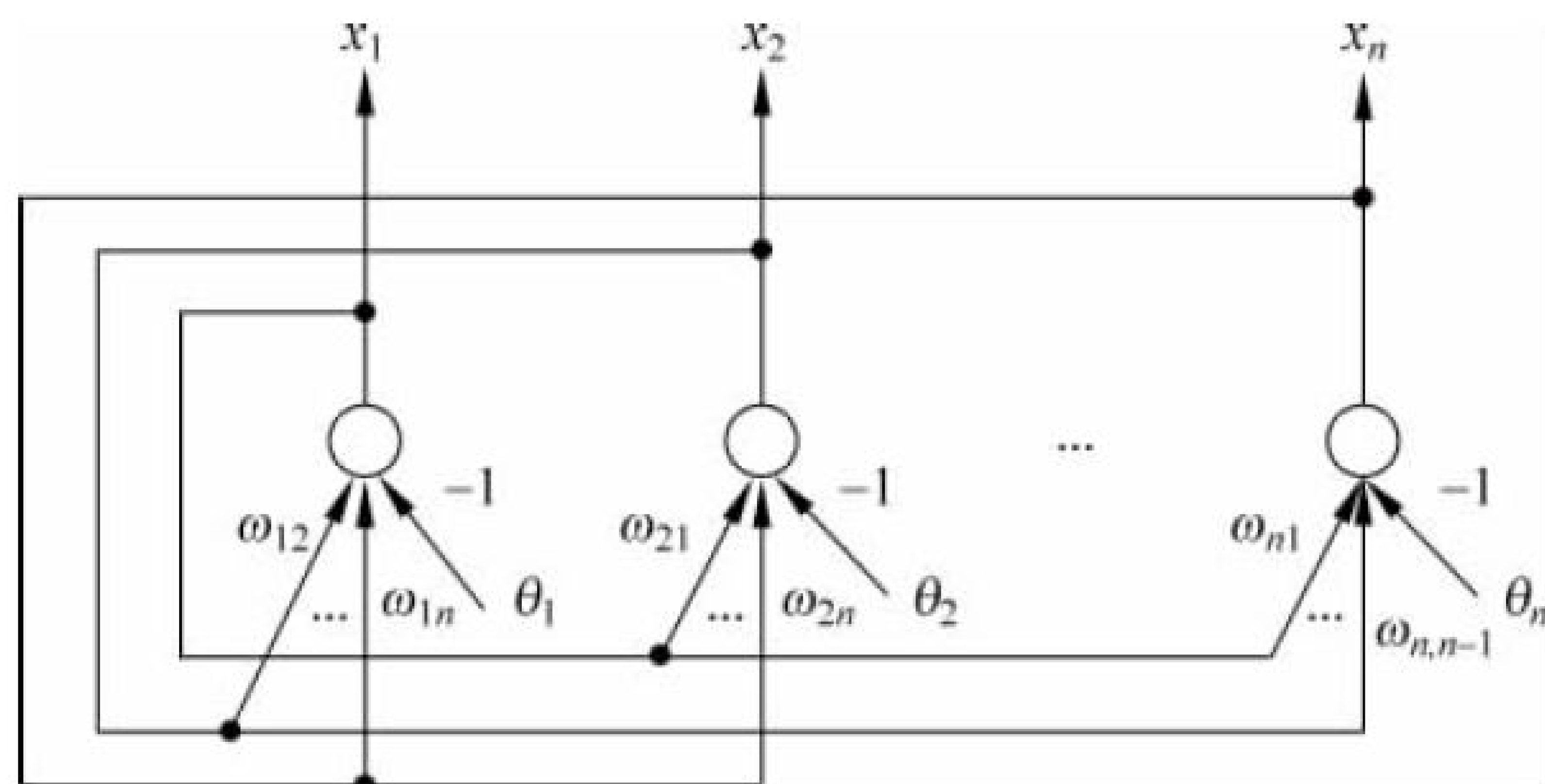


图 6-29 离散 Hopfield 网络的结构图

离散 Hopfield 网络是一个单层网络，共有 n 个神经元节点，每个节点输出均连接到其他神经元的输入，同时所有其他神经元的输出均连到该神经元的输入。对于每一个神经元节点，其工作方式仍同以前一样，即

$$s_i = \sum_{j=1,j\neq i}^{n} \omega_{ij}x_j - \theta_i \tag{6-29}$$

$$x_i = f(s_i) \tag{6-30}$$

其中，$f(\cdot)$取阶跃函数 $f(s)=\begin{cases}1, & s\geqslant 0\\ 0, & s<0\end{cases}$，或者取符号函数 $f(s)=\begin{cases}1, & s\geqslant 0\\ -1, & s<0\end{cases}$。

对于包含 n 个神经元节点的 Hopfield 网络，其网络状态是输出神经元信息的集合，由于每个输出端有两种状态，因此网络共有 2^n个状态。

如果 Hopfield 网络是稳定的，在网络的输入端加入一个输入向量，则网络的状态就会发生变化，直至网络稳定在某一特定的状态。

6.4.2 离散 Hopfield 网络的工作方式

离散 Hopfield 网络的工作方式分为同步方式和异步方式两种。

(1) 异步(串行)方式。每次只有一个神经元节点进行状态的调整计算，其他节点的状态均保持不变，即

$$x_i(k+1) = f\left(\sum_{j=1,j\neq i}^{n} \omega_{ij}x_j(k) - \theta_i\right) \tag{6-31}$$

$$x_j(k+1) = x_j(k) \tag{6-32}$$

n 个节点的调整次序可以随机选定，也可按规定的次序进行。

(2) 同步(并行)方式。所有的神经元节点同时调整状态，即

$$x_i(k+1) = f\left(\sum_{j=1,j\neq i}^{n} \omega_{ij} x_j(k) - \theta_i\right) \tag{6-33}$$

其中 $\forall i$。

该网络是动态的反馈网络，其输入是网络的状态初值：$\boldsymbol{X}(0)=[x_1(0),x_2(0),\cdots,x_n(0)]^{\mathrm{T}}$，输出是网络的稳定状态 $\lim\limits_{k\to\infty}\boldsymbol{X}(k)$。网络在异步方式下的稳定性称为异步稳定性。同理，在同步方式下的稳定性称为同步稳定性。神经网络稳定时的状态称为稳定状态。

6.4.3 离散 Hopfield 网络的稳定性和吸引子

离散 Hopfield 网络实质上是一个离散的非线性动力系统。因此，如果系统是稳定的，则它可以从任一初态收敛到一个稳定状态；若系统是不稳定的，由于网络节点输出点只有 1 和 −1(或 1 和 0)两种状态，因此系统不可能无限发散，只可能出现限幅的自持振荡或极限环。

如果将稳态视为一个记忆样本，那么初态朝稳态的收敛过程就是寻找记忆样本的过程。初态可以认为是给定样本的部分信息，网络改变的过程可以认为是部分信息找到全部信息，从而实现了联想记忆的功能。

定义 1：若网络的状态 $\boldsymbol{x}$ 满足 $\boldsymbol{x}=f(W\boldsymbol{x}-\boldsymbol{\theta})$，则称 $\boldsymbol{x}$ 为网络的稳定点或吸引子。

定理 1：对于离散 Hopfield 网络，若按异步方式调整状态，并且连接权矩阵 $\boldsymbol{W}$ 为对称矩阵，则对于任意初态，网络都最终收敛到一个吸引子。

定理 2：对于离散 Hopfield 网络，若按同步方式调整状态，并且连接权矩阵 $\boldsymbol{W}$ 为非负定对称矩阵，则对于任意初态，网络都最终收敛到一个吸引子。

由上述定理可知，对于同步方式，它对连接权矩阵 $\boldsymbol{W}$ 的要求不仅为对称矩阵，同时要求非负定。若连接权矩阵 $\boldsymbol{W}$ 不满足非负定的要求，则 Hopfield 网络可能出现自持振荡(即极限环)。比较而言，异步方式比同步方式具有更好的稳定性。但异步方式失去了神经网络并行处理的优点。

定义 2：若 $\boldsymbol{x}^{(a)}$ 是吸引子，对于异步方式，若存在一个调整次序可以从 $\boldsymbol{x}$ 演变到 $\boldsymbol{x}^{(a)}$，则称 $\boldsymbol{x}$ 弱吸引到 $\boldsymbol{x}^{(a)}$；若对于任意调整次序都可以从 $\boldsymbol{x}$ 演变到 $\boldsymbol{x}^{(a)}$，则称 $\boldsymbol{x}$ 强吸引到 $\boldsymbol{x}^{(a)}$。

定义 3：对于所有 $\boldsymbol{x}\in R(\boldsymbol{x}^{(a)})$ 均有 $\boldsymbol{x}$ 弱(强)吸引到 $\boldsymbol{x}^{(a)}$，则称 $R(\boldsymbol{x}^{(a)})$ 为 $\boldsymbol{x}^{(a)}$ 的弱(强)吸引阈。

为了保证 Hopfield 网络在异步工作时能稳定收敛，应使连接权矩阵 $\boldsymbol{W}$ 为对称矩阵，同时要求对于给定的样本必须是网络的吸引子，而且要有一定的吸引阈，这样才能正确实现联想记忆功能。要实现上述功能，通常采用 Hebb 规则来设计连接权。

设给定 m 个样本 $\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$，并设 $\boldsymbol{x}\in\{-1,1\}^n$，则按 Hebb 规则设计的连接权为

$$\omega_{ij} = \begin{cases} \sum\limits_{k=1}^{m} x_i^{(k)} x_j^{(k)}, & i\neq j \\ 0, & i=j \end{cases} \tag{6-34}$$

或

$$\begin{cases} \omega_{ij}(k) = \omega_{ij}(k-1) + x_i^{(k)} x_j^{(k)}, & k = 1,2,\cdots,m \\ \omega_{ij}(0) = 0, & \omega_{ii} = 0 \end{cases} \tag{6-35}$$

写成矩阵的形式则为

$$\boldsymbol{W} = [\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(m)}] \begin{bmatrix} \boldsymbol{x}^{(1)\mathrm{T}} \\ \boldsymbol{x}^{(2)\mathrm{T}} \\ \vdots \\ \boldsymbol{x}^{(m)\mathrm{T}} \end{bmatrix} - m\boldsymbol{I} = \sum_{k=1}^{m} \boldsymbol{x}^{(k)} \boldsymbol{x}^{(k)\mathrm{T}} - m\boldsymbol{I} = \sum_{k=1}^{m} (\boldsymbol{x}^{(k)} \boldsymbol{x}^{(k)\mathrm{T}} - \boldsymbol{I}) \tag{6-36}$$

其中，$\boldsymbol{I}$ 为单位矩阵。

当网络节点状态为 1 或 0 两种状态，即 $\boldsymbol{x} \in \{0,1\}^n$，相应的连接权为

$$\omega_{ij} = \begin{cases} \sum_{k=1}^{m} (2x_i^{(k)} - 1)(2x_j^{(k)} - 1), & i \neq j \\ 0, & i = j \end{cases} \tag{6-37}$$

或

$$\begin{cases} \omega_{ij}(k) = \omega_{ij}(k-1) + (2x_i^{(k)} - 1)(2x_j^{(k)} - 1), & k = 1,2,\cdots,m \\ \omega_{ij}(0) = 0, & \omega_{ii} = 0 \end{cases} \tag{6-38}$$

写成矩阵的形式，则

$$\boldsymbol{W} = \sum_{k=1}^{m} (2\boldsymbol{x}^{(k)} - \boldsymbol{b})(2\boldsymbol{x}^{(k)} - \boldsymbol{b})^{\mathrm{T}} - m\boldsymbol{I} \tag{6-39}$$

其中，$\boldsymbol{b} = [1,1,\cdots,1]^{\mathrm{T}}$。

6.4.4 离散 Hopfield 网络的连接权设计

Hopfield 网络的一个功能是用于联想记忆，即联想存储器。用于联想记忆时，首先通过一个学习训练过程确定网络中的权系数，使所记忆的信息在网络的 n 维超立方体的某个顶角处的能量最小。

离散 Hopfield 网络的连接权是设计出来的，设计方法的主要思路是使被记忆的模式样本对应于网络能量函数的极小值。

设有 m 个 n 维记忆模式，要设计网络连接权 ω_{ij} 和阈值 θ，使这 m 个模式正好是网络能量函数的 m 个极小值。比较常用的设计方法是"外积法"。设

$$\boldsymbol{U}_k = [\boldsymbol{U}_1^k, \boldsymbol{U}_2^k, \cdots, \boldsymbol{U}_n^k] \tag{6-40}$$

其中，$k=1,2,\cdots,m$；$U_i^k \in \{0,1\}$，$i=1,2,\cdots,n$。m 表示模式类别数；n 为每一类模式的维数；$\boldsymbol{U}_k$ 为模式 k 的向量表达。

要求网络记忆的 $m(m \leqslant n)$ 个记忆模式向量两两正交，即满足下式

$$(\boldsymbol{U}'_i)(\boldsymbol{U}_j) = \begin{cases} 0, & j \neq i \\ n, & j = i \end{cases} \tag{6-41}$$

各神经元的阈值 $\theta_i = 0$，网络的连接权矩阵按下式计算

$$\boldsymbol{W}=\sum_{k=1}^{m}\boldsymbol{U}_k(\boldsymbol{U}_k)' \tag{6-42}$$

则所有向量 $\boldsymbol{U}_k$ 在 $1\leqslant k\leqslant m$ 内都是稳定点。

在网络结构参数一定的条件下,要保证联想功能的正确实现,网络所能存储的最大的样本数与网络的节点数 n 有关。当网络结构确定时,即节点数 n 为定值时,适当地调整设计连接权可以调高网络存储的样本数。同时,对于用 Hebb 规则设计连接权的网络,如果输入样本是正交的,则可以获得最大的样本记忆数。此外,最大的样本记忆数还与吸引阈有关,吸引阈越大,则最大的样本记忆数越小。

对于网络结构参数一定的一般记忆样本而言,可以通过下述方法提高最大的样本记忆数。

设给定 m 个样本向量$\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$,先组成如下的 $n\times(m-1)$阶矩阵:

$$\boldsymbol{A}=[\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(m)},\boldsymbol{x}^{(2)}-\boldsymbol{x}^{(m)},\cdots,\boldsymbol{x}^{(m-1)}-\boldsymbol{x}^{(m)}] \tag{6-43}$$

对 $\boldsymbol{A}$ 进行奇异值分解:

$$\boldsymbol{A}=\boldsymbol{U}\sum\boldsymbol{V}^{\mathrm{T}} \tag{6-44}$$

其中,

$$\sum=\begin{bmatrix}\mathrm{S} & 0\\ 0 & 0\end{bmatrix},\quad S=\mathrm{diag}(\sigma_1,\sigma_2,\cdots,\sigma_r) \tag{6-45}$$

$\boldsymbol{U}$ 为 $n\times n$ 正交矩阵,$\boldsymbol{V}$ 为$(m-1)\times(m-1)$正交矩阵,$\boldsymbol{U}$ 可表示成

$$\boldsymbol{U}=[u_1,u_2,\cdots,u_r,u_{r+1},\cdots,u_n] \tag{6-46}$$

则 $u_1,u_2,\cdots,u_r$ 是对应于非零奇异值 $\sigma_1,\sigma_2,\cdots,\sigma_r$ 的左奇异向量,并且组成了 $\boldsymbol{A}$ 的值阈空间的正交基;$u_{r+1},\cdots,u_n$ 是 $\boldsymbol{A}$ 的值阈的正交补空间的正交基。

按如下方法组成连接权矩阵 $\boldsymbol{W}$ 和阈值向量$\boldsymbol{\theta}$。

$$\boldsymbol{W}=\sum_{k=1}^{r}u_k u_k^{\mathrm{T}} \tag{6-47}$$

$$\boldsymbol{\theta}=\boldsymbol{W}\boldsymbol{x}^{(m)}-\boldsymbol{x}^{(m)} \tag{6-48}$$

经证明,按照上述方法设计的连接权矩阵可以使得所有的样本 $\boldsymbol{x}^{(k)}$ 均为网络的吸引子。

6.4.5 离散 Hopfield 网络分类器的 MATLAB 实现

有一组三元色数据,希望将数据按照颜色数据所表征的特点,将数据按各自所属的类别归类。三元色数据如表 1-2 所示。其中,前 29 组数据已确定类别,后 30 组数据待确定类别。

1. 运用 Hopfield 网络的步骤

将具体数据的分类标准作为网络的标准模式使网络记忆它们的特征,得到权值,也就是得到一个 Hopfield 网络的结构;输入采样点的实测值,利用得到的网络进行联想,最后确定采样点属于哪种标准模式,就可以得到分类结果。运用 Hopfield 网络进行分类的步骤如下:

(1) 设定网络的记忆模式,即将预存储的模式或类别进行编码,得到取值为 1 和 −1 的记忆模式。由于原始给定数据分为 4 类,采用了 3 项特征来进行判别,因此记忆模式为

$$U_k = [u_1^k, u_2^k, \cdots, u_n^k] \tag{6-49}$$

其中，$k=1,2,\cdots,n$；$n=40$。用1来表示达到某一分级标准，用-1表示未达到某一分级标准，表6-2所列为将数据标准化且压缩在$\{-1,1\}$后进行的数据离散化和类别编码。

表 6-2　数据离散化和类别编码

分　类	特　征　1				特　征　2				特　征　3			
1类	1	−1	−1	−1	−1	1	1	−1	−1	−1	−1	1
2类	−1	1	1	−1	−1	−1	−1	1	1	−1	−1	−1
3类	1	1	1	1	1	1	1	1	1	−1	−1	1
4类	1	−1	−1	−1	−1	−1	−1	1	−1	−1	1	−1
特征类	−1	−0.5	0.5	1	−1	−0.5	0.5	1	−1	−0.5	0.5	1

表6-2中的-0.5和0.5是指在$-1\sim-0.5$和$0.5\sim1$的特征指标。

(2) 建立网络，即运用MATLAB工具箱提供的newhop函数建立Hopfield网络，参数为U_k，并且可得到设计权值矩阵$\boldsymbol{W}$及阈值向量$\boldsymbol{\theta}$。

(3) 将待分类的数据转化为网络的欲识别模式，即转化为二值型的模式。

(4) 将其设为网络的初始状态，运用MATLAB提供的sim函数进行多次迭代使其收敛。最终得出所属类别。

综上所述，Hopfield网络的分类器设计过程如图6-30所示。

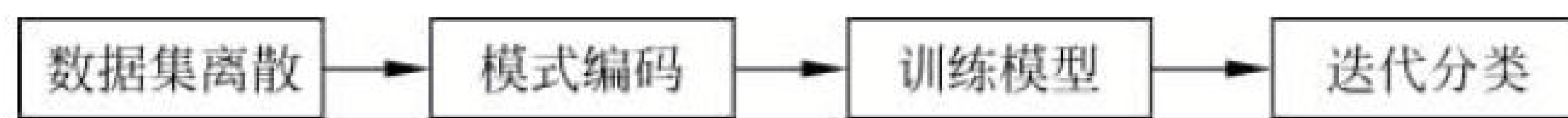


图6-30　Hopfield网络的分类器设计过程

其中的关键步骤是数据集离散化和模式编码，分类器的性能好坏基本由这几步决定。尤其是分辨率的高低，很大程度上依赖离散化和模式编码的好坏。

2. 数据集离散化

数据离散化的目的是定义一组映射，允许在各种抽象级别上处理数据，在多个层面上发现知识。常用的数据集离散化方法有分箱、直方图分析、聚类分析和基于熵的数据离散化。

为了将取值控制在一个合理的范围内，将监测特征参量的值域变化范围划分间隔，称为箱。通过将数据分布到不同的箱中，并利用箱中数据的均值或中位数替换箱中的每个值，实现数据离散化。常用的分箱策略有：等宽分箱，这种方法中每个分箱的间隔相同；等高分箱，每个分箱所包含的元组相同；基于同质分箱，这种方法中每个分箱的大小是基于相应方向中的元组分布相似进行划分的。

直方图离散化是指属性A的直方图将A的数据取值分布划分为不相交的子集或桶，这些子集或桶沿水平轴显示，其高度或面积与该桶所代表的平均出现频率成正比。通常每个桶代表某个属性的一段连续值。

聚类技术将数据视为对象，通过聚类分析所获得的组或类有如下性质：同一组或类中的对象彼此相似，而不同组或类中的对象彼此不相似。

基于熵的数据离散化是通过递归地划分数值属性，使之分层离散化。

Hopfield网络的数据离散化采用分箱与直方图结合的方法，如图6-31所示。选出数据

集相同属性的最大值与最小值，差值通过直方图和等宽分箱的方法得到。Hopfield 神经网络中每个节点的输出只有两种状态{−1 或+1}，因此，要将特征量转化成为数据矩阵，存储于网络中。其中，白色区域表示+1，黑色区域表示−1。

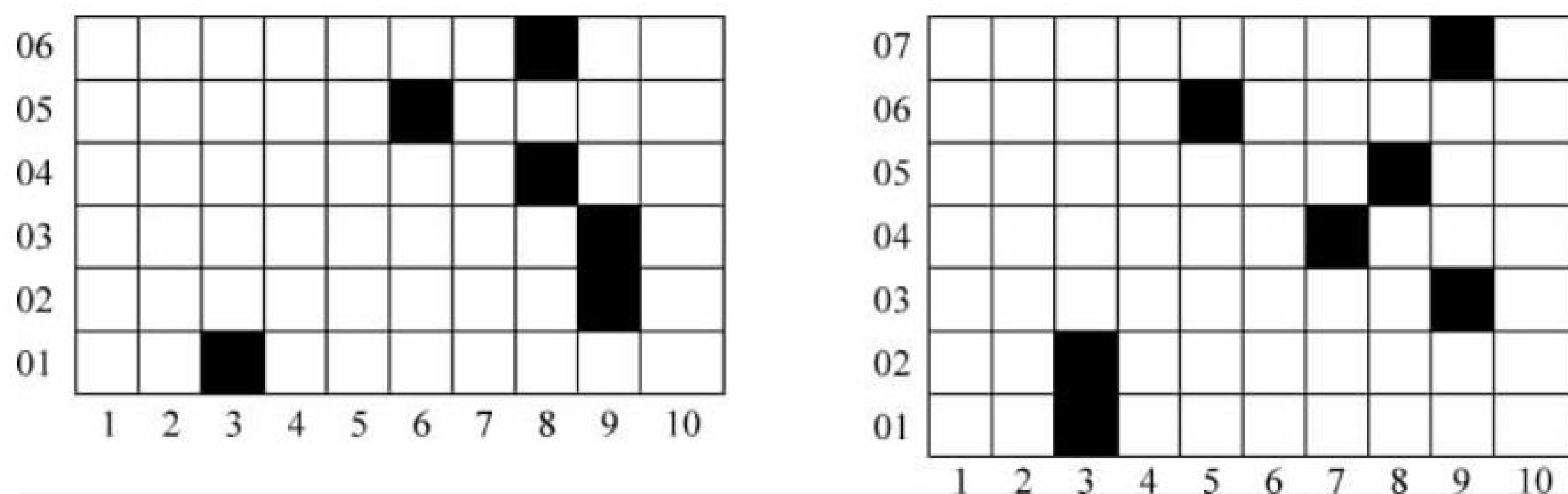


图 6-31 利用等宽分箱和直方图的特征离散表示

在本例中，利用分箱与直方图结合的方法将数据离散化。首先将数据存放到数据文件 data_sample. dat 中，数据内容及格式如图 6-32 所示。

```
data_sample - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1507.13 1556.89 1954.51
343.07  3271.72 2036.94
2201.94 3196.22 935.53
2232.43 3077.87 1298.87
1580.1  1752.07 2463.04
1962.4  1594.97 1835.95
1495.18 1957.44 3498.02
1125.17 1594.39 2937.73
24.22   3447.31 2145.01
1269.07 1910.72 2701.97
1802.07 1725.81 1966.35
1817.36 1927.4  2328.79
1860.45 1782.88 1875.13
```

图 6-32 data_sample. dat 数据内容及格式

MATLAB 程序代码如下：

```
clear;
clc;
p = importdata('data_sample.dat');
 % 数据标准化同时压缩在{ - 1,1}
[pn,minp,maxp] = premnmx(p);
P = zeros(59,4);
for i = 1:59
    if pn(i,1) == - 1
        P(i,1) = 1;P(i,2) = - 1;P(i,3) = - 1;P(i,4) = - 1;
    else if (1 - pn(i,1))> 1
            P(i,1) = - 1;P(i,2) = 1; P(i,3) = - 1;P(i,4) = - 1;
        else if pn(i,1) == 1
            P(i,1) = - 1;P(i,2) = - 1; P(i,3) = - 1;P(i,4) = 1;
            else P(i,1) = - 1;P(i,2) = - 1; P(i,3) = 1;P(i,4) = - 1;
            end
        end
    end
end
```

```
%利用以上循环将第一个特征向量离散化
P1 = P;
P = zeros(59,4);
for i = 1:59
    if pn(i,2) == -1
        P(i,1) = 1;P(i,2) = -1;P(i,3) = -1;P(i,4) = -1;
    else if (1 - pn(i,2))>1
            P(i,1) = -1;P(i,2) = 1; P(i,3) = -1;P(i,4) = -1;
        else if pn(i,2) == 1
            P(i,1) = -1;P(i,2) = -1; P(i,3) = -1;P(i,4) = 1;
            else  P(i,1) = -1;P(i,2) = -1; P(i,3) = 1;P(i,4) = -1;
            end
        end
    end
end
P2 = P;
P = zeros(59,4);
for i = 1:59
    if pn(i,3) == -1
        P(i,1) = 1;P(i,2) = -1;P(i,3) = -1;P(i,4) = -1;
    else if (1 - pn(i,3))>1
            P(i,1) = -1;P(i,2) = 1; P(i,3) = -1;P(i,4) = -1;
        else if pn(i,3) == 1
            P(i,1) = -1;P(i,2) = -1; P(i,3) = -1;P(i,4) = 1;
            else P(i,1) = -1;P(i,2) = -1; P(i,3) = 1;P(i,4) = -1;
            end
        end
    end
end
P3 = P;
%输出离散化的数据
P = [P1,P2,P3]
```

运行上述程序后，即可得到数据离散化结果：

```
P =
  -1   1  -1  -1   1  -1  -1  -1  -1  -1  -1   1
   1  -1  -1  -1  -1  -1  -1   1  -1  -1   1  -1
  -1  -1  -1   1  -1  -1   1  -1   1  -1  -1  -1
   1  -1  -1  -1  -1   1  -1  -1  -1  -1  -1   1
   1  -1  -1  -1  -1  -1  -1   1  -1  -1   1  -1
   1  -1  -1  -1  -1  -1   1  -1  -1  -1  -1   1
  -1   1  -1  -1   1  -1  -1  -1  -1  -1  -1   1
  -1  -1   1  -1  -1  -1  -1   1   1  -1  -1  -1
   1  -1  -1  -1  -1  -1  -1   1  -1  -1   1  -1
   1  -1  -1  -1  -1  -1  -1   1  -1  -1   1  -1
   1  -1  -1  -1  -1  -1   1  -1  -1  -1  -1   1
   1  -1  -1  -1  -1  -1  -1   1  -1  -1   1  -1
   1  -1  -1  -1  -1  -1  -1   1  -1  -1   1  -1
```

```
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
    -1     1    -1    -1    -1    -1    -1     1     1    -1    -1    -1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
    -1    -1     1    -1     1    -1    -1    -1    -1    -1    -1     1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
    -1     1    -1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
    -1     1    -1    -1     1    -1    -1    -1    -1    -1    -1     1
    -1     1    -1    -1     1    -1    -1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1    -1     1    -1    -1    -1    -1     1
    -1    -1    -1     1    -1     1    -1    -1     1    -1    -1    -1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
    -1     1    -1    -1     1    -1    -1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
     1    -1    -1    -1    -1    -1     1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1    -1     1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1    -1    -1     1    -1     1    -1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
    -1    -1     1    -1    -1    -1    -1     1     1    -1    -1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
    -1    -1    -1     1     1    -1    -1    -1    -1    -1     1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1    -1    -1     1    -1    -1     1    -1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
    -1     1    -1    -1     1    -1    -1    -1    -1    -1    -1     1
     1    -1    -1    -1    -1     1    -1    -1    -1    -1    -1     1
    -1    -1     1    -1     1    -1    -1    -1    -1    -1    -1     1
    -1    -1     1    -1     1    -1    -1    -1    -1    -1    -1     1
```

3. 模式编码

按照表 6-2 进行模式编码，MATLAB 程序代码如下：

```
one = [1 -1 -1 -1 -1 1 1 -1 -1 -1 -1 1];
two = [-1 1 1 -1 -1 -1 -1 1 1 -1 -1 -1];
three = [1 1 1 1 1 1 1 1 1 -1 -1 1];
four = [1 -1 -1 -1 -1 -1 -1 1 -1 -1 1 - 1];
```

4. 网络学习

Hopfield 网络学习的 MATLAB 程序代码如下：

```
T = [one;two;three;four]';
%输出 Hopfield 网络
net = newhop(T);
%输出权值与偏差
w = net.lw{1,1},b = net.b{1};
```

5. 输出网络分类结果

输出网络分类结果的 MATLAB 程序代码如下：

```
L = zeros(59,1);
for i = 1:59
a = {[P(i,1),P(i,2),P(i,3),P(i,4),P(i,5),P(i,6),P(i,7),P(i,8),P(i,9),P(i,10),P(i,11),
P(i,12)]'};
[y,Pf,Af] = sim(net,{1 60},[],a);
    if y{50}' == one
        L(i,1) = 1;
    else if y{50}' == two
            L(i,1) = 2;
        else if y{50}' == four
                L(i,1) = 4;
            else L(i,1) = 3;
            end
        end
    end
end
L
```

6. 以图形方式输出分类结果

以图形方式输出分类结果的 MATLAB 程序代码如下：

```
hold off
f = L';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
plot3(p(:,1),p(:,2),p(:,3),'o');
line(p(index1,1),p(index1,2),p(index1,3),'linestyle','none','marker','*','color','g');
line(p(index2,1),p(index2,2),p(index2,3),'linestyle','none','marker','*','color','r');
line(p(index3,1),p(index3,2),p(index3,3),'linestyle','none','marker','+','color','b');
line(p(index4,1),p(index4,2),p(index4,3),'linestyle','none','marker','+','color','y');
box;grid on;hold on;
xlabel('A');
ylabel('B');
zlabel('C');
title('Hopfield Network State Space');
```

运行程序后，系统分类结果如图 6-33 所示。

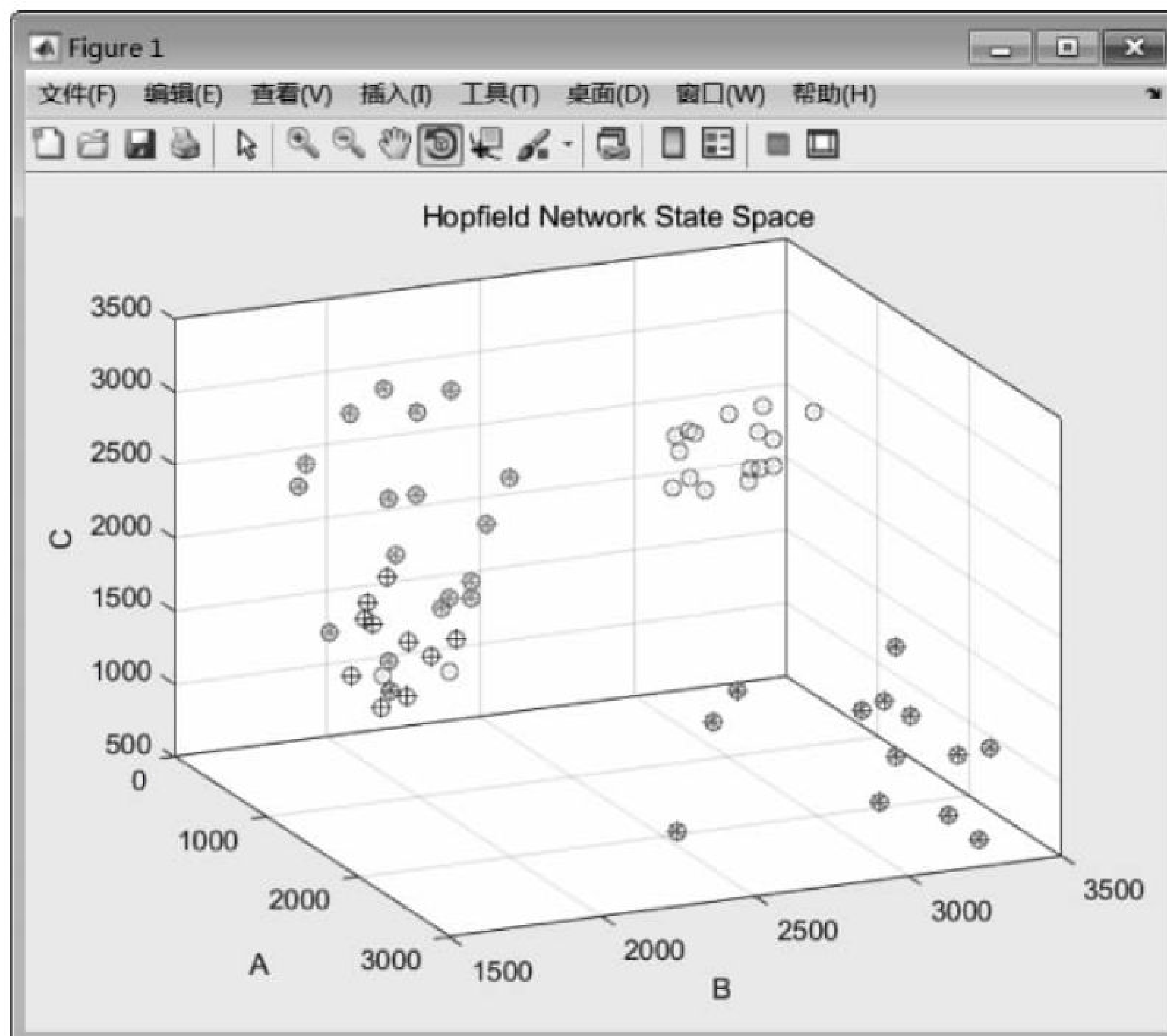


图 6-33　系统分类结果

将 Hopfield 网络的分类结果与原始数据的分类结果进行对照，结果如表 6-3 所示。

表 6-3 Hopfield 网络的分类结果与原始数据的分类结果的对照

序 号	A	B	C	原始分类结果	Hopfield 网络分类结果
1	1739.94	1675.15	2395.96	3	3
2	373.3	3087.05	2429.47	4	4
3	1756.77	1652	1514.98	3	3
4	864.45	1647.31	2665.9	1	1
5	222.85	3059.54	2002.33	4	4
6	877.88	2031.66	3071.18	1	1
7	1803.58	1583.12	2163.05	3	3
8	2352.12	2557.04	1411.53	2	2
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
11	1571.17	1731.04	1735.33	3	1
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
16	1418.79	1775.89	2772.9	1	1
17	1845.59	1918.81	2226.49	3	1
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
20	1692.62	1867.5	2108.97	3	1
21	1680.67	1575.78	1725.1	3	3
22	2802.88	3017.11	1984.98	2	2
23	172.78	3084.49	2328.65	4	4
24	2063.54	3199.76	1257.21	2	2
25	1449.58	1641.58	3405.12	1	1
26	1651.52	1713.28	1570.38	3	2
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4

6.4.6 结论

Hopfield 网络具有很强的自组织、自学习能力。其采用模式联想的方式运作，网络回想时间很短，一般只需一到两次迭代即可完成，且既适用于定量指标的分类参数也适用于定性指标的分类参数，参数越多，评价结果越可靠，运算结果直接给出样本应属于的酒瓶类别。因此，有其独特的优越性。但由于网络结构和输入方式的局限，其应用于酒瓶颜色分类结果的精度受到一定的影响，需要具体的改进才能准确分类，还有待于进一步发展完善。

6.5 径向基函数

从结构上分类，神经网络可分为前馈神经网络和反馈神经网络，而从对函数的逼近功能上分类，神经网络可分为全局逼近和局部逼近。如果网络的一个或多个连接权系数或自适应可调参数在输入空间的每一点对任何一个输入都有影响，则称该网络为全局逼近网络；若对输入空间的某个局部区域，只有少数几个连接权影响网络的输出，则称该网络为局部逼近网络。当只有少量的连接权需要进行调整，从而使局部逼近网络具有学习速度快的优点。径向基函数(radial basis function，RBF)就属于局部逼近神经网络。

6.5.1 径向基函数的网络结构及工作方式

径向基函数 RBF 神经网络(简称径向基网络)是由 J. Moody 和 C. Darken 于 20 世纪 80 年代末提出的一种神经网络结构，RBF 神经网络是一种性能良好的前向网络，具有最佳逼近及克服局部极小值问题的性能。RBF 神经网络起源于数值分析中的多变量插值的径向基函数方法，径向基函数的网络结构如图 6-34 所示。

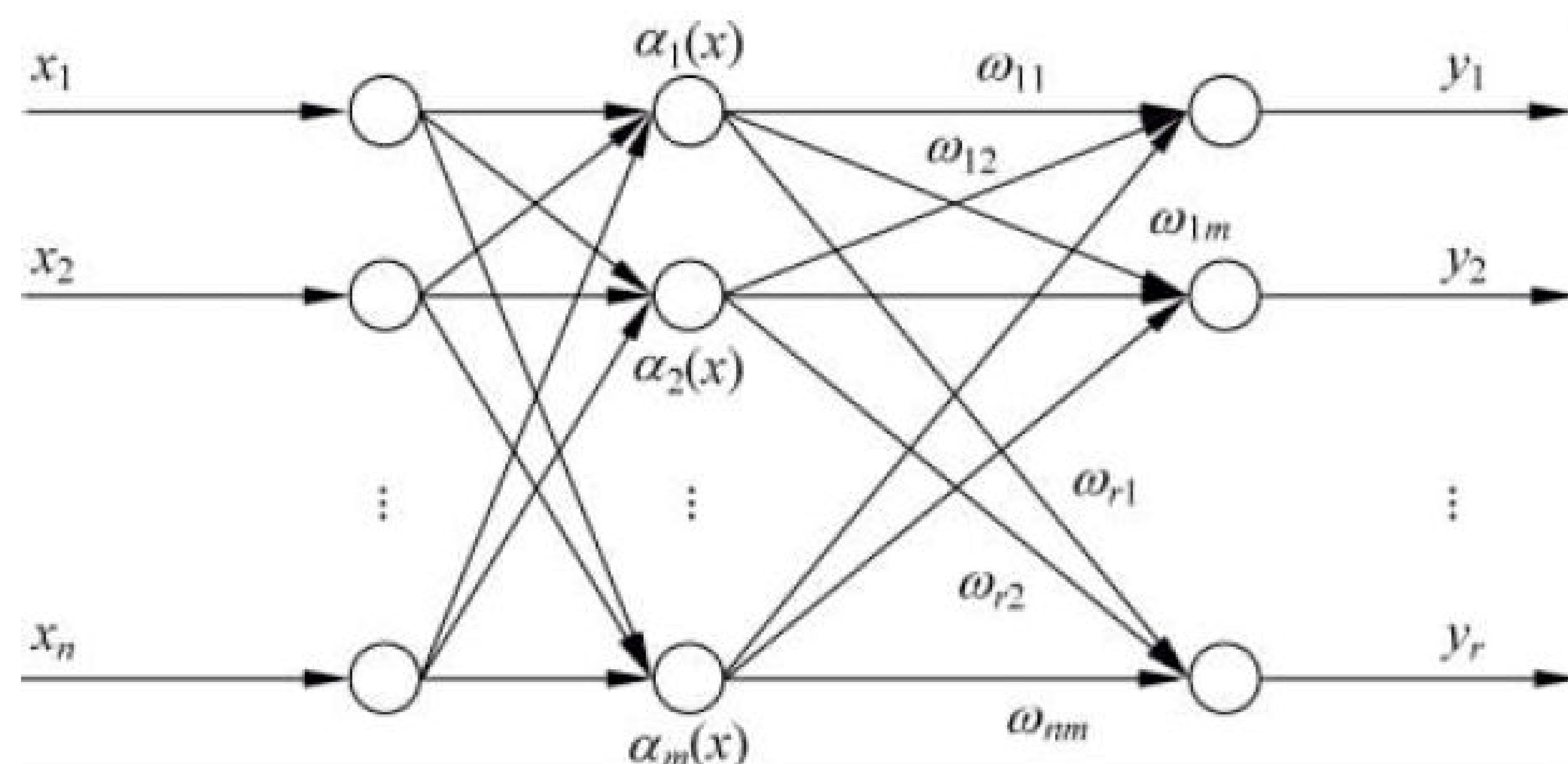


图 6-34　径向基函数的网络结构

RBF 神经网络的拓扑结构是一种三层前向网络：输入层由信号源节点构成，仅起到数据信息的传递作用，对输入信息不进行任何变换；第二层为隐含层，节点数视需要而定，隐含层神经元的核函数(作用函数)为高斯函数，对输入信息进行空间映射变换；第三层为输出层，它对输入模式做出响应，输出层神经元的作用函数为线性函数，对隐含层神经元输出的信息进行线性加权后输出，作为整个神经网络的输出结果。

RBF 神经网络只有一个隐含层，隐含层单元采用径向基函数 $\alpha_j(x)$ 作为其输出特性，输入层到隐含层之间的权值均固定为 1；输出节点为线性求和单元，隐含层到输出节点之间的权值 w_{ij} 可调，因此输出为

$$\boldsymbol{y}_i = \sum_{j=1}^{m} \omega_{ij}\alpha_j(x), \quad i = 1,2,\cdots,r \tag{6-50}$$

径向基函数为某种沿径向对称的标量函数。隐含层径向基神经元模型结构如图 6-35 所示。由图 6-35 可见，径向基网络传递函数是以输入向量与阈值向量之间的距离 $\| X - C_j \|$

作为自变量的，其中 $\| X-C_j \|$ 是通过输入向量和加权矩阵 $\boldsymbol{C}$ 的行向量的乘积得到的。径向基网络传递函数可以取多种形式，最常用的有下面三种：

（1）Gaussian 函数

$$\Phi_i(t) = \mathrm{e}^{-\frac{t^2}{\delta_i^2}} \tag{6-51}$$

（2）Reflected sigmoidal 函数

$$\Phi_i(t) = \frac{1}{1+\mathrm{e}^{\frac{t^2}{\delta^2}}} \tag{6-52}$$

（3）逆 Multiquadric 函数

$$\Phi_i(t) = \frac{1}{(t^2+\delta_i^2)^a}, \quad a>0 \tag{6-53}$$

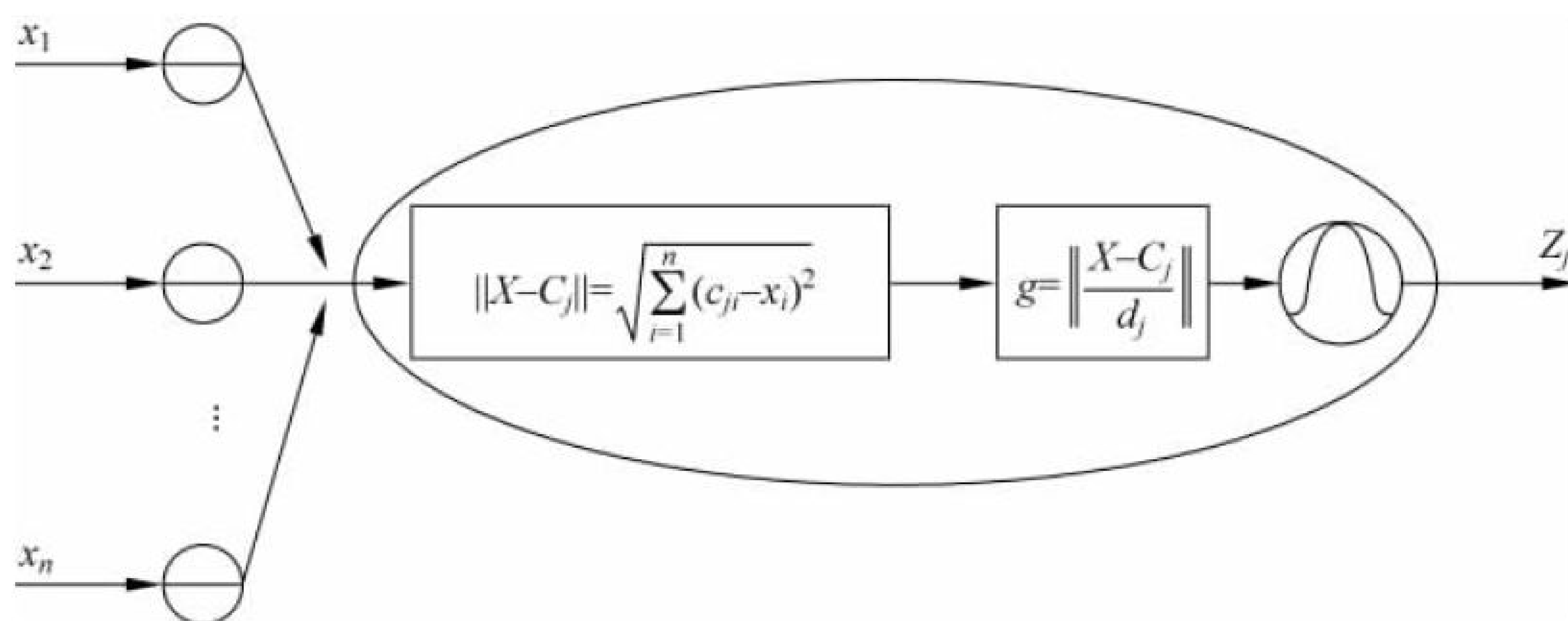


图 6-35 径向基神经元模型结构

最常用的 RBF 基函数是高斯基函数

$$\alpha_j(x) = \psi_j(\| x-c_j \| / \sigma_j) = \mathrm{e}^{-\frac{\| x-c_j \|^2}{\sigma_j^2}} \tag{6-54}$$

其中，c_j 是第 j 个基函数的中心点，σ_j 是一个可以自由选择的参数，它决定了该基函数围绕中心点的宽度，控制了函数的作用范围。基于高斯基函数的 RBF 神经网络的拓扑结构如图 6-36 所示。

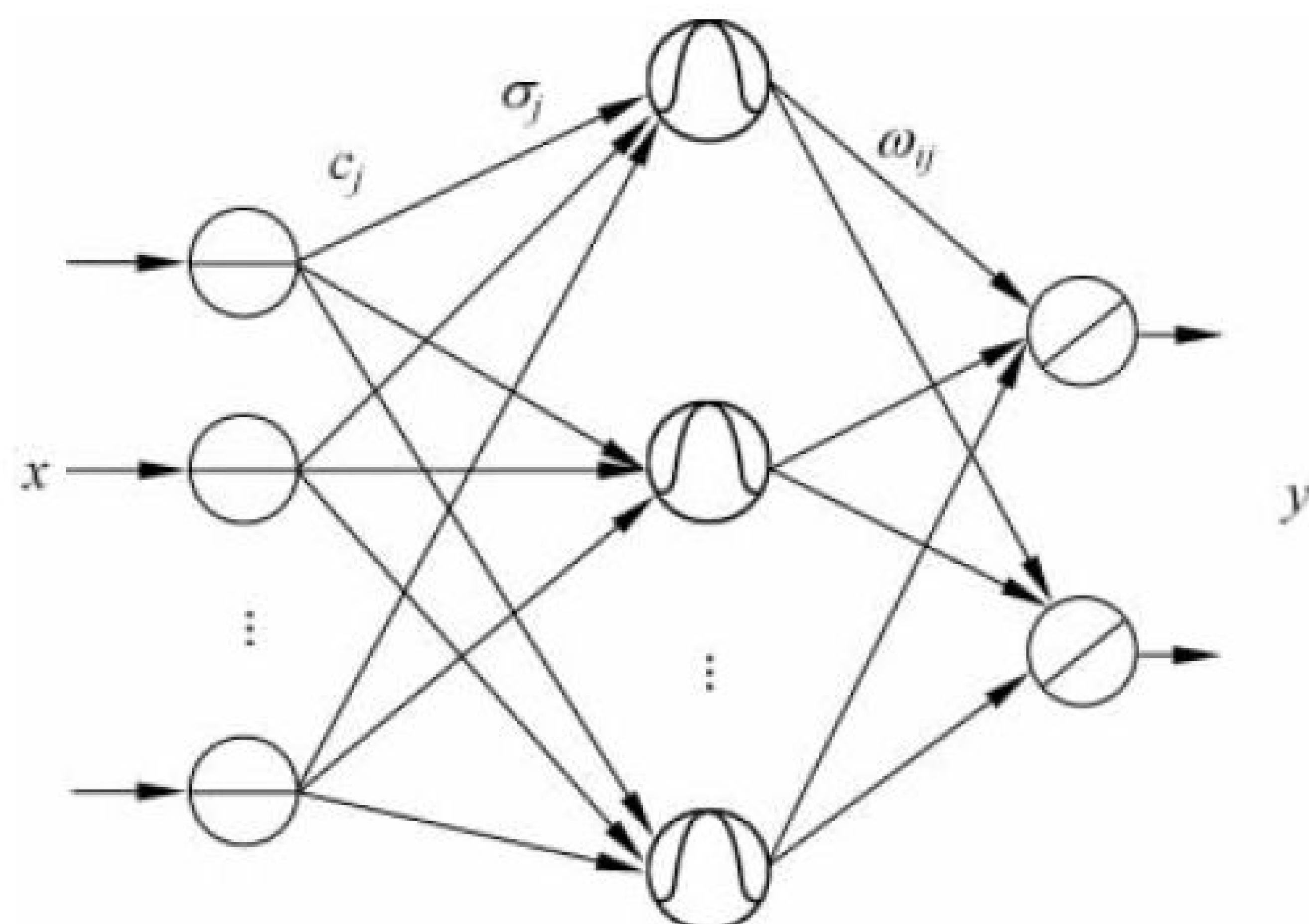


图 6-36 基于高斯基函数的 RBF 神经网络的拓扑结构

其连接权的学习算法为

$$\omega_{ij}(l+1) = \omega_{ij}(l) + \beta[y_i^d - y_i(l)]\alpha_j(x)/\alpha^{\mathrm{T}}(x)\alpha(x) \tag{6-55}$$

当输入自变量为0时,传递函数取得最大值为1。随着权值和输入向量的距离不断减小,网络输出递增。也就是说,径向基函数对输入信号在局部产生响应。函数的输入信号 x 靠近函数的中央范围时,隐含层节点将产生较大的输出。

当将输入向量添加到网络输入端时,径向基层每个神经元都会输出一个值,这个值代表输入向量与神经元权值向量之间的接近程度。如果输入向量与权值向量相差很多,则径向基层输出接近0,经过第二层的线性神经元,输出也接近于0;如果输入向量与权值向量很接近,则径向基层的输出接近于1,经过第二层的线性神经元输出值就靠近第二层权值。在这个过程中,如果只有一个径向基神经元的输出为1,而其他的神经元输出均为0或者接近0,那么线性神经元的输出就相当于输出为1的神经元相对应的第二层权值的值。一般情况下,不止一个神经元的输出为1,所以输出值也就会有不同。

6.5.2 径向基函数网络的特点及作用

径向基函数由于采用了高斯基函数,具有如下优点:

(1) 表示形式简单,即使是多变量输入也不增加太多的复杂性。

(2) 径向对称。

(3) 光滑性好。

径向基函数网络具有如下作用:

(1) 一般任何函数都可以表示成一组基函数的加权和,因此径向基函数网络可以逼近任意未知函数。

(2) 在径向基网络中,从输入层到隐含层的基函数是一种非线性映射,而输出则是线性映射。因此,径向基函数可以看成是将原始的非线性可分的特征空间变换到另一个高维空间。通过合理选择这一变换,使在新的空间中原问题线性可分。

6.5.3 径向基函数网络参数选择

径向基函数网络中,可调整第 j 个基函数的中心点 c_j 及其方差 σ_j。常采用如下方法进行调整。

(1) 根据经验选择函数中心点 c_j。如果只训练样本的分布能代表所给问题,则可根据经验选定均匀的 m 各个中心点,其间距为 d,则基函数方差 $\sigma_j = d/\sqrt{2m}$。

(2) 用聚类方法选择基函数。可以以各类聚类中心作为基函数的中心点,而以各类样本的方差的某一函数作为各个基函数的宽度参数。

6.5.4 RBF 网络分类器的 MATLAB 实现

以表1-2所示的三元色数据为例,希望将数据按照颜色数据所表征的特点,将数据按各自所属的类别归类。其中,前29组数据已确定类别,后30组数据待确定类别。

1. 从样本数据库中获取训练数据

取前 29 组数据作为训练样本。为了编程方便,先对这 29 组数据按类别进行升序排序。重新排序后的数据如表 6-4 所示。

表 6-4 重新排序后的数据表

序 号	A	B	C	分类结果
4	864.45	1647.31	2665.9	1
6	877.88	2031.66	3071.18	1
16	1418.79	1775.89	2772.9	1
25	1449.58	1641.58	3405.12	1
8	2352.12	2557.04	1411.53	2
14	2297.28	3340.14	535.62	2
15	2092.62	3177.21	584.32	2
18	2205.36	3243.74	1202.69	2
19	2949.16	3244.44	662.42	2
22	2802.88	3017.11	1984.98	2
24	2063.54	3199.76	1257.21	2
1	1739.94	1675.15	2395.96	3
3	1756.77	1652	1514.98	3
7	1803.58	1583.12	2163.05	3
11	1571.17	1731.04	1735.33	3
17	1845.59	1918.81	2226.49	3
20	1692.62	1867.5	2108.97	3
21	1680.67	1575.78	1725.1	3
26	1651.52	1713.28	1570.38	3
2	373.3	3087.05	2429.47	4
5	222.85	3059.54	2002.33	4
9	401.3	3259.94	2150.98	4
10	363.34	3477.95	2462.86	4
12	104.8	3389.83	2421.83	4
13	499.85	3305.75	2196.22	4
23	172.78	3084.49	2328.65	4
27	341.59	3076.62	2438.63	4
28	291.02	3095.68	2088.95	4
29	237.63	3077.78	2251.96	4

将排序后的数据及其类别绘制在三维图中直观地表示出来,作为 RBF 网络训练时应达到的目标。排序后的数据及其类别的三维图如图 6-37 所示。

将样本数据及分类结果分别存放到.dat 文件中。数据文件内容及格式如图 6-38 所示。

2. 设置径向基函数的分布密度

Spread 为径向基层的分布密度,又称散布常数,默认值为 1。散布常数是 RBF 网络设计过程中一个非常重要的参数。一般情况下,散布常数应该足够大,使得神经元响应区域覆



图 6-37 排序后的数据及其类别的三维图



(a) rbf_train_sample_data.dat文件内容及格式



(b) rbf_train_target_data.dat文件内容及格式



(c) rbf_simulate_data.dat文件内容及格式

图 6-38 数据文件内容及格式

盖所有输入区间。

3. 调用 newrb 构建并训练径向基函数神经网络

在 MATLAB 中,构建径向基函数网络的函数有两个,分别为 newrbe()函数和 newrb()函数。应用 newrbe()函数可以快速设计一个径向基函数网络,并且使得设计误差为 0,调用代码如下:

```
net = newrbe(p,t,spread);
```

其中，p 为输入向量；t 为期望输出向量(目标值)，SPREAD 为径向基层的散布常数，默认值为 1。输出为一个径向基网络，其权值和阈值完全满足输入和期望值关系要求。

由 newrbe()函数构建的径向基函数网络，其径向基层(第一层)神经元数目等于输入向量的个数，那么在输入向量较多的情况下，则需要很多的神经元，这就给网络设计带来一定的难度。函数 newrb()则可自动增加网络的隐含层神经元数目，直到均方差满足精度或神经元数目达到最大为止。

newrb 定义为 net＝newrb(p,t,GOAL,SPREAD,MN,DF)，各个参数的定义如下。

P：Q 个输入向量的 R×Q 维矩阵。这里 Q＝29，R＝3。

T：Q 个目标类别向量的 S×Q 维矩阵。这里 S＝1。

GOAL：期望的均方误差值，默认为 0.0。这里选择默认值。

SPREAD：径向基函数的散布常数，默认为 1.0。

MN：神经元的最大数目，默认等于 Q。这里设置为 28。

DF：每次显示时增加的神经元数目，默认为 25，并且返回一个新的径向基函数网络。这里设置为 2。

4. 调用 sim 及识别样本

调用 sim，测试 RBF 网络的训练效果；再次调用 sim 识别样本所属类别。基于 MATLAB 的 RBF 模式分类程序代码如下：

```
clear;
clc;
%网络训练目标
pConvert = importdata('C:\Users\Administrator\Desktop\RBF\rbf_train_sample_data.dat');
p = pConvert';
t = importdata('C:\Users\Administrator\Desktop\RBF\rbf_train_target_data.dat');
plot3(p(1,:),p(2,:),p(3,:),'o');
grid;box;
for i = 1:29,text(p(1,i),p(2,i),p(3,i),sprintf(' %g',t(i))),end
hold off
f = t';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
line(p(1,index1),p(2,index1),p(3,index1),'linestyle','none','marker',' * ','color','g');
line(p(1,index2),p(2,index2),p(3,index2),'linestyle','none','marker',' * ','color','r');
line(p(1,index3),p(2,index3),p(3,index3),'linestyle','none','marker',' + ','color','b');
line(p(1,index4),p(2,index4),p(3,index4),'linestyle','none','marker',' + ','color','y');
box;grid on;hold on;
axis([0 3500 0 3500 0 3500]);
title('训练用样本及其类别');
xlabel('A');
ylabel('B');
zlabel('C');
```

```
% RBF 网络的创建和训练过程
net = newrb(p,t,0,410,28,2);
A = sim(net,p)
plot3(p(1,:),p(2,:),p(3,:),'r .'),grid;box;
axis([0 3500 0 3500 0 3500])
for i = 1:29,text(p(1,i),p(2,i),p(3,i),sprintf(' % g',A(i))),end
hold off
f = A';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
line(p(1,index1),p(2,index1),p(3,index1),'linestyle','none','marker',' * ','color','g');
line(p(1,index2),p(2,index2),p(3,index2),'linestyle','none','marker',' * ','color','r');
line(p(1,index3),p(2,index3),p(3,index3),'linestyle','none','marker',' + ','color','b');
line(p(1,index4),p(2,index4),p(3,index4),'linestyle','none','marker',' + ','color','y');
box;grid on;hold on;
title('网络训练结果');
xlabel('A');
ylabel('B');
zlabel('C');
% 对测试样本进行分类
pConvert = importdata('C:\Users\Administrator\Desktop\RBF\rbf_simulate_data.dat');
p = pConvert';
a = sim(net,p)
```

运行程序后,系统首先输出训练样本及其类别分类图,如图 6-39 所示。接着输出 RBF 网络的训练结果图,如图 6-40 所示。

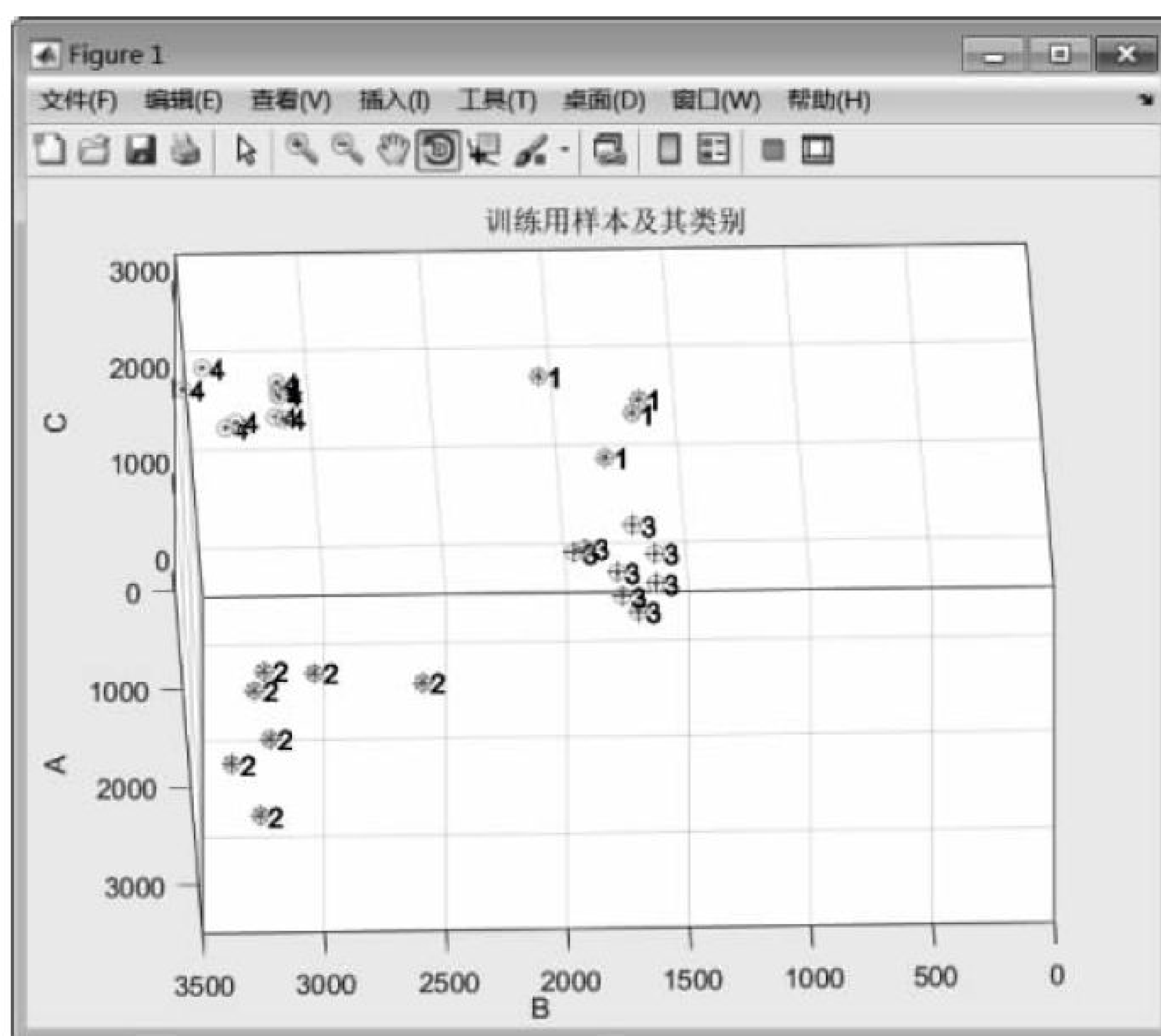


图 6-39 训练用样本及其类别分类图

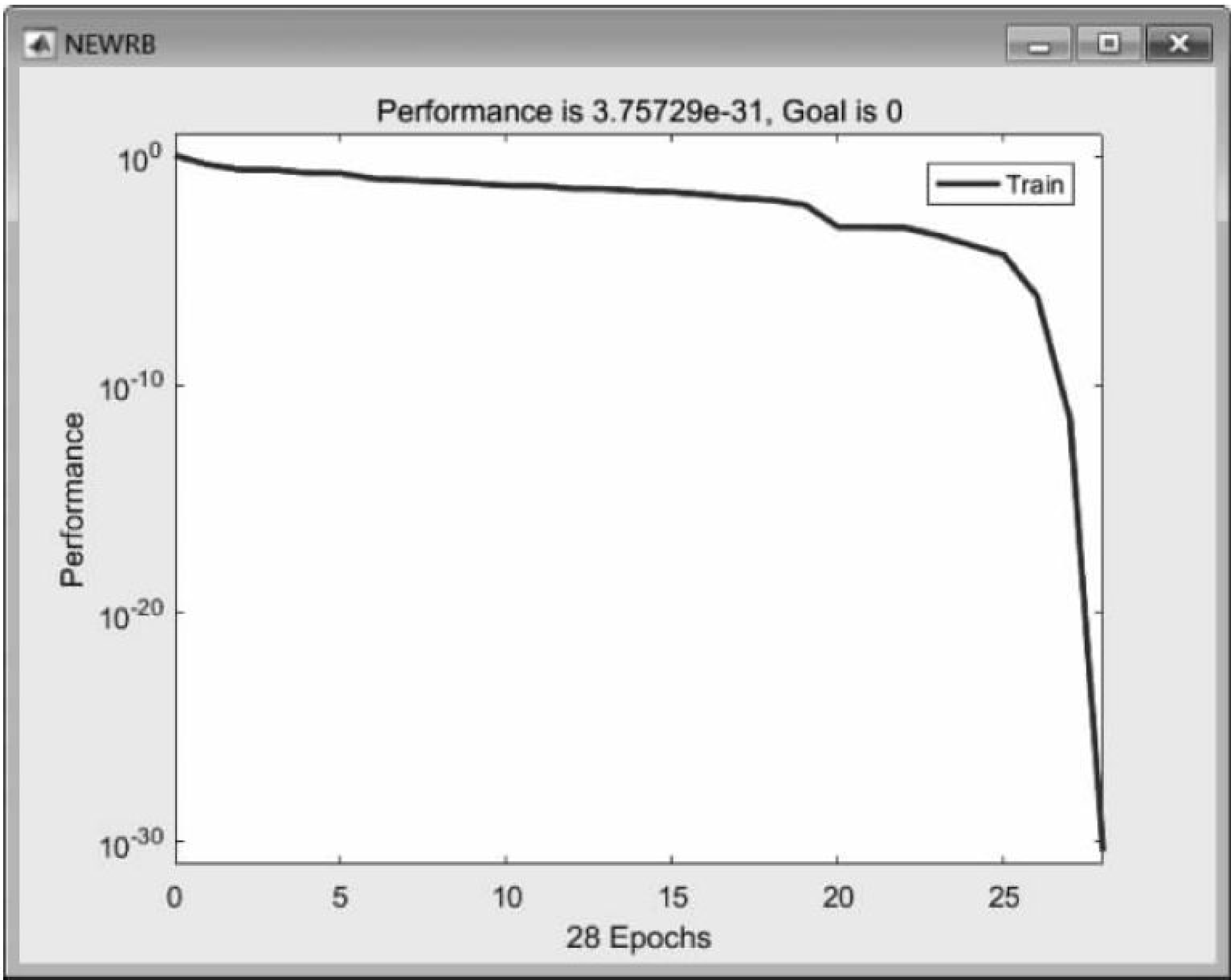


图 6-40 RBF 网络的训练结果图

训练后的 RBF 网络对训练数据进行分类后的结果与目标结果对比，如表 6-5 所示。

表 6-5 训练后的 RBF 网络对训练数据进行分类后的结果与目标结果对比表

序 号	A	B	C	目标结果	RBF 网络分类结果
4	864.45	1647.31	2665.9	1	1
6	877.88	2031.66	3071.18	1	1
16	1418.79	1775.89	2772.9	1	1
25	1449.58	1641.58	3405.12	1	1
8	2352.12	2557.04	1411.53	2	2
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
22	2802.88	3017.11	1984.98	2	2
24	2063.54	3199.76	1257.21	2	2
1	1739.94	1675.15	2395.96	3	3
3	1756.77	1652	1514.98	3	3
7	1803.58	1583.12	2163.05	3	3
11	1571.17	1731.04	1735.33	3	3
17	1845.59	1918.81	2226.49	3	3
20	1692.62	1867.5	2108.97	3	3
21	1680.67	1575.78	1725.1	3	3
26	1651.52	1713.28	1570.38	3	3
2	373.3	3087.05	2429.47	4	4
5	222.85	3059.54	2002.33	4	4

续表

序　号	A	B	C	目标结果	RBF 网络分类结果
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
23	172.78	3084.49	2328.65	4	4
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4

训练后的 RBF 网络对训练数据进行分类后的结果与目标结果完全吻合，可见 RBF 网络训练效果良好。以下为神经元逐渐增加的过程及对应输出的均方误差。

```
NEWRB,neurons = 0,MSE = 1.1082
NEWRB,neurons = 2,MSE = 0.262521
NEWRB,neurons = 4,MSE = 0.188316
NEWRB,neurons = 6,MSE = 0.104082
NEWRB,neurons = 8,MSE = 0.0794035
NEWRB,neurons = 10,MSE = 0.0524248
NEWRB,neurons = 12,MSE = 0.0377437
NEWRB,neurons = 14,MSE = 0.0302773
NEWRB,neurons = 16,MSE = 0.0209541
NEWRB,neurons = 18,MSE = 0.0124128
NEWRB,neurons = 20,MSE = 0.000818943
NEWRB,neurons = 22,MSE = 0.000771163
NEWRB,neurons = 24,MSE = 0.000131081
NEWRB,neurons = 26,MSE = 7.66274e-07
NEWRB,neurons = 28,MSE = 3.75729e-31
```

从运行过程可以看出，随着神经元数目的逐渐增加，均方误差逐渐减小。当神经元数目增加到 28 时，误差已经很接近 0，基本可以达到要求。

继续执行程序，系统将给出训练后的 RBF 网络对训练样本数据的识别结果图，如图 6-41 所示。

继续执行程序，可得到测试样本的分类结果：

```
a =
1 ～ 9 列
1.0000    1.0000    1.0000    1.0000    2.0000    2.0000    2.0000    2.0000    2.0000
10 ～ 18 列
2.0000    2.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000
19 ～ 27 列
3.0000    4.0000    4.0000    4.0000    4.0000    4.0000    4.0000    4.0000    4.0000
28 ～ 36 列
4.0000    4.0000    2.8969    3.2124    2.9232    4.0000    2.2147    2.4485    3.0550
37 ～ 45 列
4.0009    2.6013    3.1286    3.1476    1.3241    2.1283    4.0008    3.6605    3.9999
46 ～ 48 列
3.1801    3.9996    1.8306
```

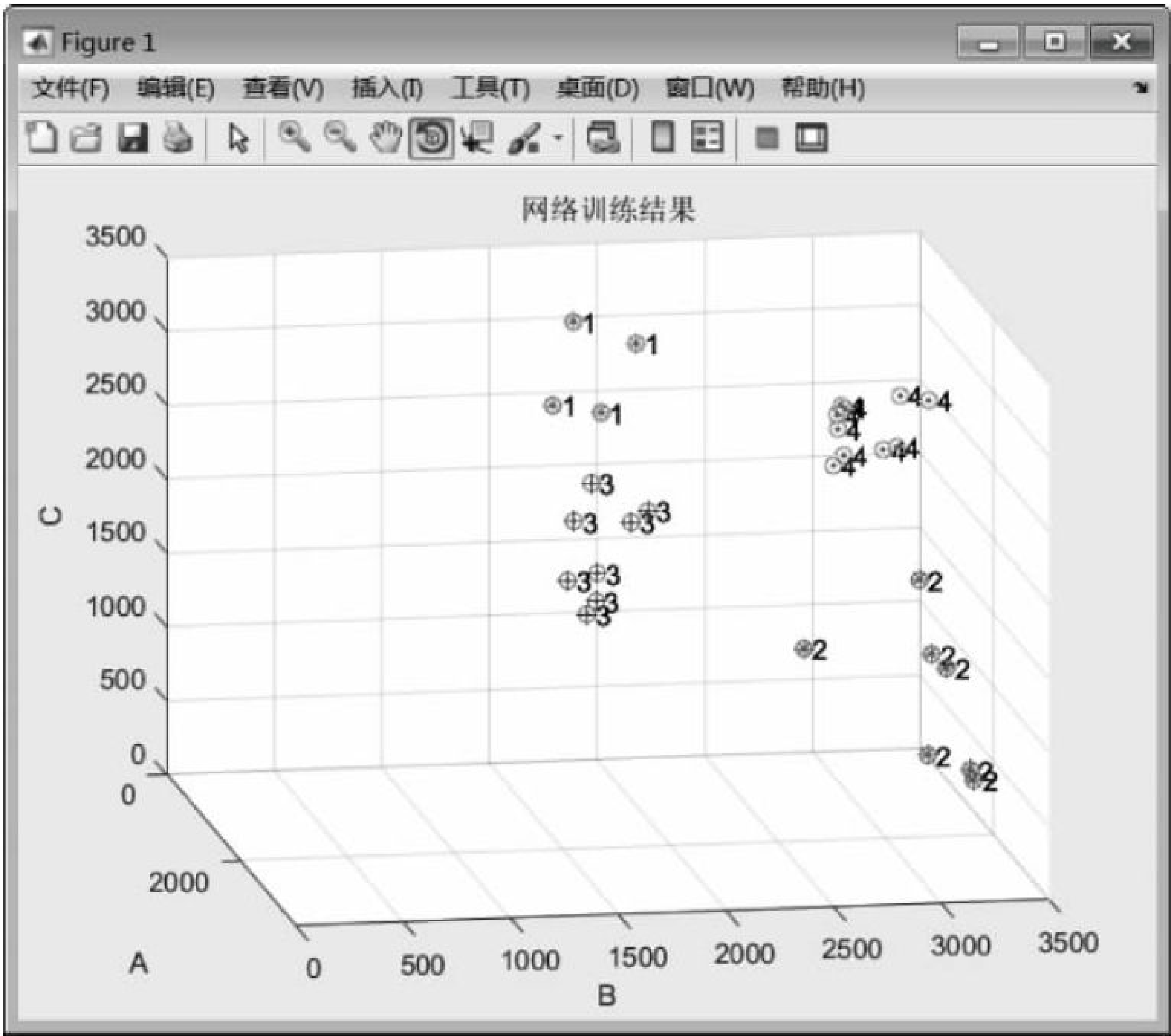


图 6-41 训练后的 RBF 网络对训练样本数据的识别结果图

a 为测试数据的分类结果，对 a 进行近似处理后可得最终的分类结果：

```
a =
1 ～ 9 列
1       1       1       1       2       2       2       2       2
10 ～ 18 列
2       2       3       3       3       3       3       3       3
19 ～ 27 列
3       4       4       4       4       4       4       4       4
28 ～ 36 列
4       4       3       3       3       4       2       2.4485  3
37 ～ 45 列
4       2.6013  3       3       1       2       4       3.6605  4
46 ～ 48 列
3       4       2
```

6.5.5 结论

由酒瓶颜色分类实例可以发现，在未对数据进行近似处理之前，类别均为小数。而且对于某些数据，分类的结果介于两类之间，无法人为决定其所属类别。这是因为，其一，虽然目前已经证明径向基函数网络能够以任意精度逼近任意连续函数，但对于本例的离散数据，理论上就不能做到完全逼近；其二，径向基函数神经网络的输出层为线性层，神经元层的输出乘以输出层权值之后直接输出结果，输出层不会计算某一数据属于某一类别的概率。由径

向基函数神经元与竞争神经元一起构成的另一种神经网络结构——概率神经网络(PNN)可以解决这个问题。

6.6 广义回归神经网络

广义回归神经网络(generalized regression neural network,GRNN)是径向基神经网络的一种。GRNN具有很强的非线性映射能力和柔性网络结构以及高度的容错性和健壮性,适用于解决非线性问题。GRNN在逼近能力和学习速度上较RBF网络有更强的优势,网络最后收敛于样本量积聚较多的优化回归面,并且在样本数据较少时,预测效果也较好。此外,网络还可以处理不稳定的数据。因此,GRNN在信号分析、结构分析、教育产业、能源、食品科学、控制决策系统、药物设计、金融领域、生物工程等各个领域得到了广泛应用。

6.6.1 GRNN的结构

GRNN在结构上与RBF网络较为相似。它是四层结构,如图6-42所示,分别为输入层(input layer)、模式层(pattern layer)、求和层(summation layer)和输出层(output layer)。对应网络输入 *net*=newgrnn(*P*,*T*,*SPREAD*),其输出为 *Y*=sim(*net*,*P*)。

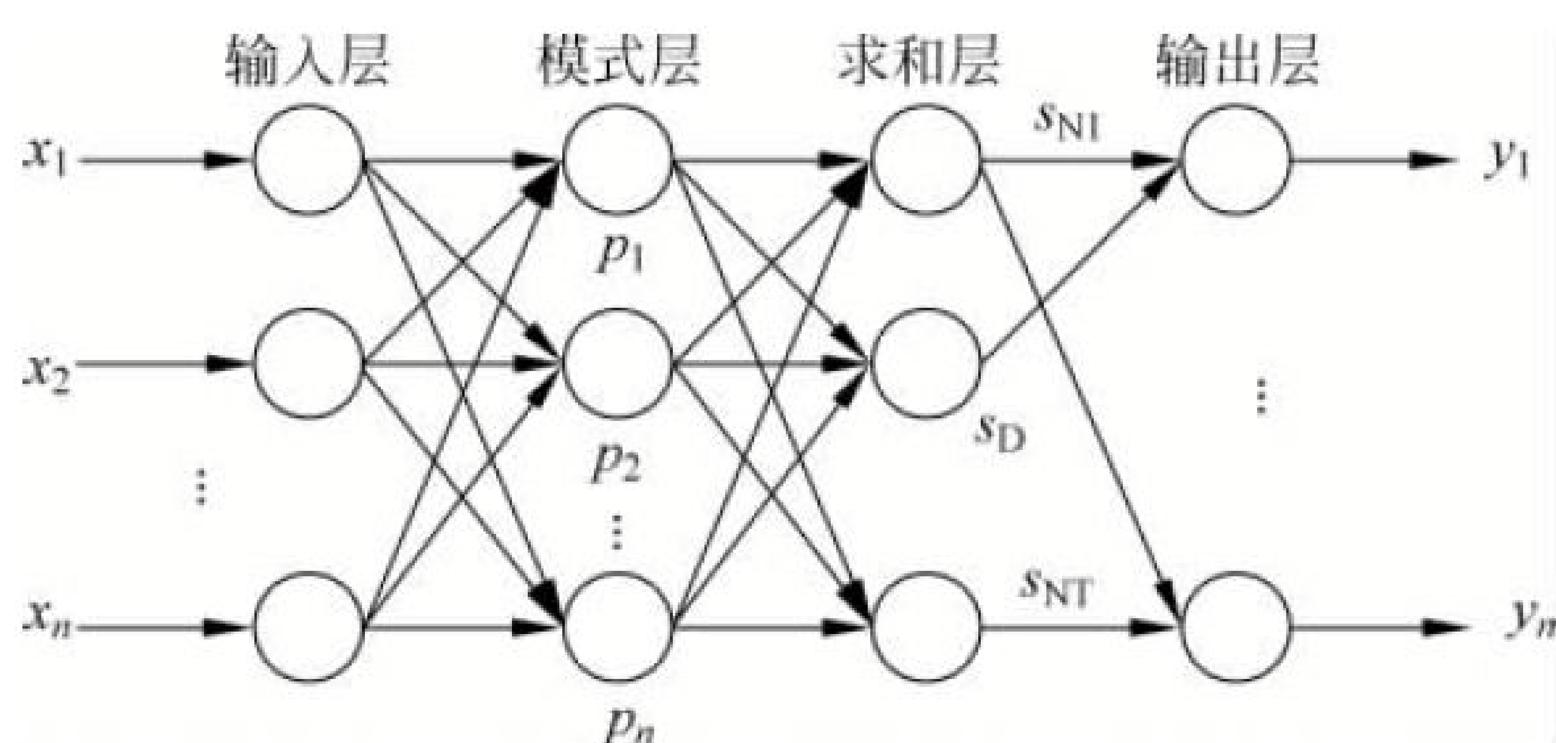


图6-42 广义回归神经网络结构图

1. 输入层

输入层神经元的数目等于学习样本中输入向量的维数,各神经元是简单的分布单元,直接将输入变量传递给模式层。

2. 模式层

模式层神经元数目等于学习样本的数目 n,各神经元对应不同的样本,模式层神经元传递函数为

$$p_i = \exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right], \quad i=1,2,\cdots,n \tag{6-56}$$

神经元 i 的输出为输入变量与其对应的样本 X 之间 Euclid 距离平方的指数平方 $D_i^2=(X-X_i)^{\mathrm{T}}(X-X_i)$ 的指数形式。式中,X 为网络输入变量;X_i 为第 i 个神经元对应的学习样本。

3. 求和层

求和层中使用两种类型神经元进行求和。

一类计算公式为 $\sum_{i=1}^{n}\exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right]$，它对所有模式层神经元的输出进行算术求和，其模式层与各神经元的连接权值为1，传递函数为

$$S_D=\sum_{j=1}^{n}p_j \tag{6-57}$$

另一类计算公式为 $\sum_{i=1}^{n}Y_j\exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right]$，它对所有模式的神经元进行加权求和，模式层中第 i 个神经元与求和层第 j 个分子求和，神经元之间的连接权值为第 i 个输出样本 Y_i 中的第 j 个元素，传递函数为

$$S_{nj}=\sum_{n=1}^{n}y_{nj}P_i,\quad j=1,2,\cdots,k \tag{6-58}$$

4. 输出层

输出层中的神经元数目等于学习样本中输出向量的维数 k，各神经元将求和层的输出相除，神经元 j 的输出对应估计结果 $\hat{Y}(X)$的第 j 个元素，即

$$y_j=\frac{S_{nj}}{S_D},\quad j=1,2,\cdots,k \tag{6-59}$$

6.6.2 GRNN 的理论基础

GRNN 的理论基础是非线性回归分析，非独立变量 Y 相对于独立变量 x 的回归分析实际上是计算具有最大概率值的 y。设随机变量 x 和随机变量 y 的联合概率密度函数为 $f(x,y)$，已知 x 的观测值为 X，则 y 相对于 X 的回归，即条件均值为

$$\hat{Y}=E(y/X)=\frac{\int_{-\infty}^{+\infty}yf(X,y)\mathrm{d}y}{\int_{-\infty}^{+\infty}f(X,y)\mathrm{d}y} \tag{6-60}$$

$\hat{Y}$ 即为在输入为 X 的条件下，Y 的预测输出。

应用 Parzen 非参数估计，可由样本数据集 $\{x_i,y_i\}_{i=1}^{n}$，估算密度函数 $\hat{f}(X,y)$。

$$\hat{f}(X,y)=\frac{1}{n(2\pi)^{\frac{p-1}{2}}\delta^{p+1}}\sum_{i=1}^{n}\exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right]\exp\left[-\frac{(X-Y_i)^2}{2\delta^2}\right] \tag{6-61}$$

式中，X_i，Y_i 为随机变量 x 和 y 的样本观测值；n 为样本容量；p 为随机变量 x 的维数；δ 为高斯函数的宽度系数，在此称为光滑因子。

用 $\hat{f}(X,y)$代替 $f(X,y)$代入式(6-60)，并交换积分与加和的顺序

$$\hat{Y}(X)=\frac{\sum_{i=1}^{n}\exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right]\int_{-\infty}^{+\infty}y\exp\left[-\frac{(Y-Y_i)^2}{2\delta^2}\right]\mathrm{d}y}{\sum_{i=1}^{n}\exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right]\int_{-\infty}^{+\infty}\exp\left[-\frac{(Y-Y_i)^2}{2\delta^2}\right]\mathrm{d}y} \tag{6-62}$$

由于 $\int_{-\infty}^{+\infty} z e^{e^z} dz = 0$，对两个积分进行计算后可得网络的输出 $\widehat{Y}(X)$ 为

$$\widehat{Y}(X) = \frac{\sum_{i=1}^{n} Y_i \exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right]}{\sum_{i=1}^{n} \exp\left[-\frac{(X-X_i)^{\mathrm{T}}(X-X_i)}{2\delta^2}\right]} \tag{6-63}$$

估计值 $\widehat{Y}(X)$ 为所有样本观测值 Y_i 的加权平均，每个观测值 Y_i 的权重因子为相应的样本 Y_i 与 X 之间 Euclid 距离平方的指数。当光滑因子 δ 非常大的时候，$\widehat{Y}(X)$ 近似于所有样本因变量的均值。相反，当光滑因子 δ 趋于 0 的时候，$\widehat{Y}(X)$ 和训练样本非常接近，当需要预测的点被包含在训练样本集中时，公式求出的因变量的预测值会和样本中对应的因变量非常接近，而一旦碰到样本中未能包含进去的点，有可能预测效果会非常差，这种现象说明网络的泛化能力差。当 δ 取值适中，求解预测值 $\widehat{Y}(X)$ 时，所有训练样本的因变量都被考虑了进去，与预测点距离近的样本点对应的因变量被加了更大的权。

6.6.3 GRNN 的特点及作用

相比 BP 网络，GRNN 具有以下优点：

(1) GRNN 同样能够以任意精度逼近任意非线性连续函数，且预测效果接近甚至优于 BP 网络。

(2) GRNN 的训练非常简单。当训练样本通过隐含层的同时，网络训练随即完成。它的训练过程不需要迭代，因此较 BP 网络的训练过程快得多，更适合于在线数据的实时处理。

(3) GRNN 所需的训练样本较 BP 网络少得多。要取得同样的效果，GRNN 所需样本是 BP 网络的 1%。

(4) GRNN 的结构相对简单，除了输入和输出层外，一般只有两个隐含层，即模式层和求和层。而模式中隐含单元的个数，与训练样本的个数是相同的。

(5) 由于网络结构简单，因此不需要对网络的隐含层数和隐含单元的个数进行估算和猜测。由于它是从径向基函数引申而来，因此只有一个自由参数，即径向基函数的平滑参数。而它的优化值可以通过交叉验证的方法非常容易得到。

6.6.4 GRNN 分类器的 MATLAB 实现

将表 1-2 所示数据按照颜色数据所表征的特点，按各自所属类别归类。其中，前 29 组数据已确定类别，后 30 组数据待确定类别。

1. 从样本数据库中获取训练数据

取前 29 组数据作为训练样本。并将样本数据及分类结果分别存放到“.dat”文件中。数据文件内容及格式如图 6-43 所示。

(a) SelfOrganization.dat文件内容及格式

(b) SelfOrganizationtarget.dat文件内容及格式　　(c) SelfOrganizationSimulation.dat文件内容及格式

图 6-43　数据文件内容及格式

2. 设置径向基函数的分布密度

Spread 为径向基层的分布密度，又称散布常数，默认值为 1。散布常数是 GRNN 网络设计过程中一个非常重要的参数。一般情况下，散布常数应该足够大，使得神经元响应区域能够覆盖所有输入区间。

3. 调用 newgrnn 构建并训练广义神经网络

在 MATLAB 中，应用 newgrnn()函数可以快速设计一个广义神经网络，并且使得设计误差为 0，调用代码如下：

```
net = newgrnn(p,t,spread);
```

其中，p 为输入向量；t 为期望输出向量(目标值)，spread 为广义神经网络的散布常数，默认值为 1。输出为一个广义神经网络，其权值和阈值完全满足输入和期望值关系要求。

4. 调用 sim 及识别样本

调用 sim，测试 GRNN 的训练效果；再次调用 sim 识别样本所属类别。基于 MATLAB 的 GRNN 模式分类程序代码如下：

```
clear;
clc;
```

```
%网络训练样本
pConvert = importdata('C:\Users\Administrator\Desktop\ln\SelfOrganizationtrain.dat');;
p = pConvert';
%训练样本的目标矩阵
t = importdata('C:\Users\Administrator\Desktop\ln\SelfOrganizationtarget.dat');
plot3(p(1,:),p(2,:),p(3,:),'o');
grid;box;
for i = 1:29,text(p(1,i),p(2,i),p(3,i),sprintf(' %g',t(i))),end
hold off
f = t;
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
line(p(1,index1),p(2,index1),p(3,index1),'linestyle','none','marker',' * ','color','g');
line(p(1,index2),p(2,index2),p(3,index2),'linestyle','none','marker',' * ','color','r');
line(p(1,index3),p(2,index3),p(3,index3),'linestyle','none','marker',' + ','color','b');
line(p(1,index4),p(2,index4),p(3,index4),'linestyle','none','marker',' + ','color','y');
box;grid on;hold on;
axis([0 3500 0 3500 0 3500]);
title('训练用样本及其类别');
xlabel('A');
ylabel('B');
zlabel('C');
t = t';
t = ind2vec(t);
spread = 30;
% GRNN网络的创建和训练过程
net = newgrnn(p,t,spread);
A = sim(net,p);
Ac = vec2ind(A)
plot3(p(1,:),p(2,:),p(3,:),'.'),grid;box;
axis([0 3500 0 3500 0 3500])
for i = 1:29,text(p(1,i),p(2,i),p(3,i),sprintf(' %g',Ac(i))),end
%以图形方式输出训练结果
hold off
f = Ac';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
line(p(1,index1),p(2,index1),p(3,index1),'linestyle','none','marker',' * ','color','g');
line(p(1,index2),p(2,index2),p(3,index2),'linestyle','none','marker',' * ','color','r');
line(p(1,index3),p(2,index3),p(3,index3),'linestyle','none','marker',' + ','color','b');
line(p(1,index4),p(2,index4),p(3,index4),'linestyle','none','marker',' + ','color','y');
box;grid on;hold on;
title('网络训练结果');
xlabel('A');
ylabel('B');
zlabel('C');
```

```
%对待分类样本进行分类
pConvert = importdata('C:\Users\Administrator\Desktop\ln\SelfOrganizationSimulation.dat');
p = pConvert';
a = sim(net,p);
ac = vec2ind(a)
```

运行程序后,系统首先输出训练用样本及其类别分类图,如图 6-44 所示。接着输出 GRNN 的训练结果图,如图 6-45 所示。

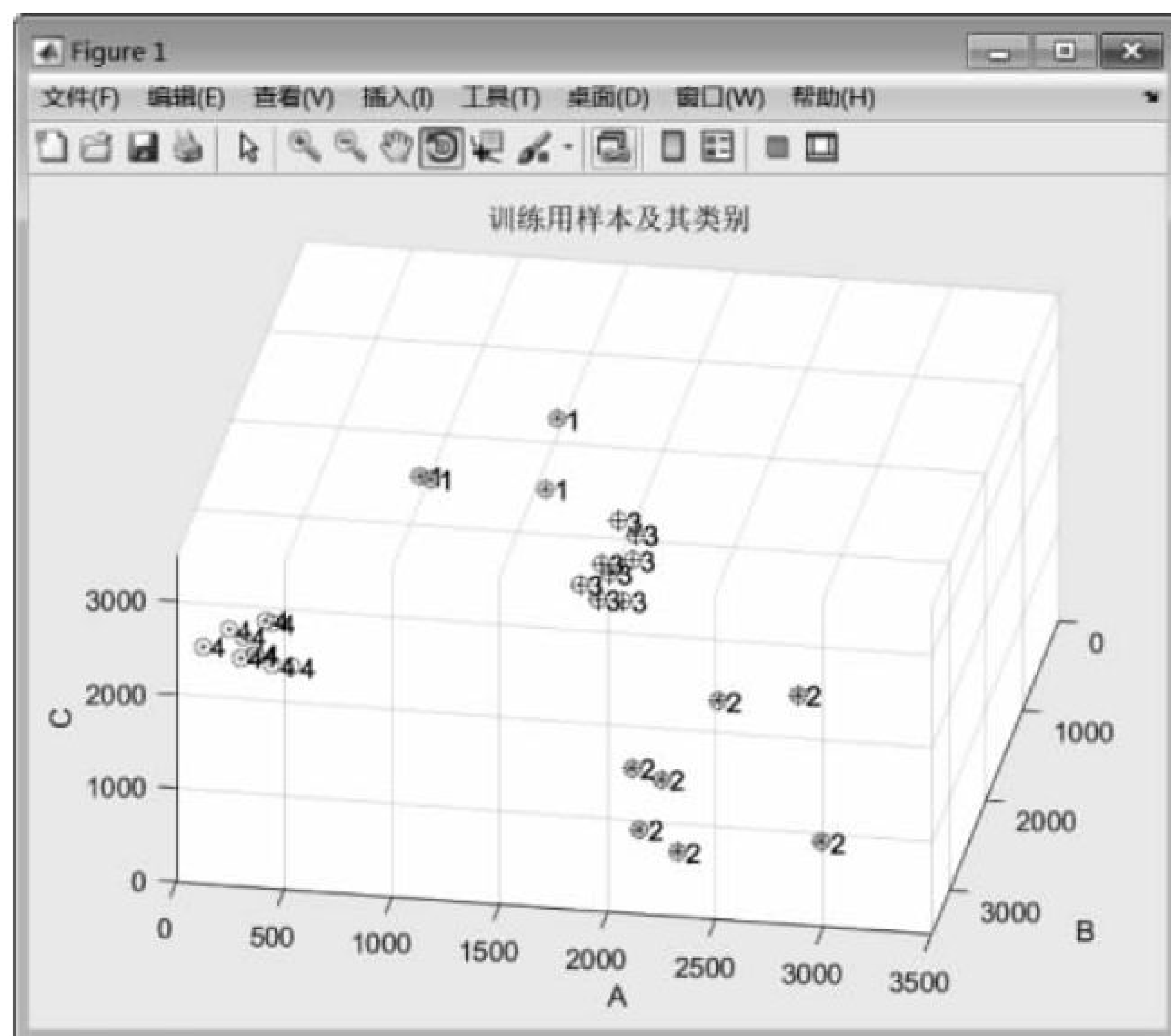


图 6-44　训练用样本及其类别分类图

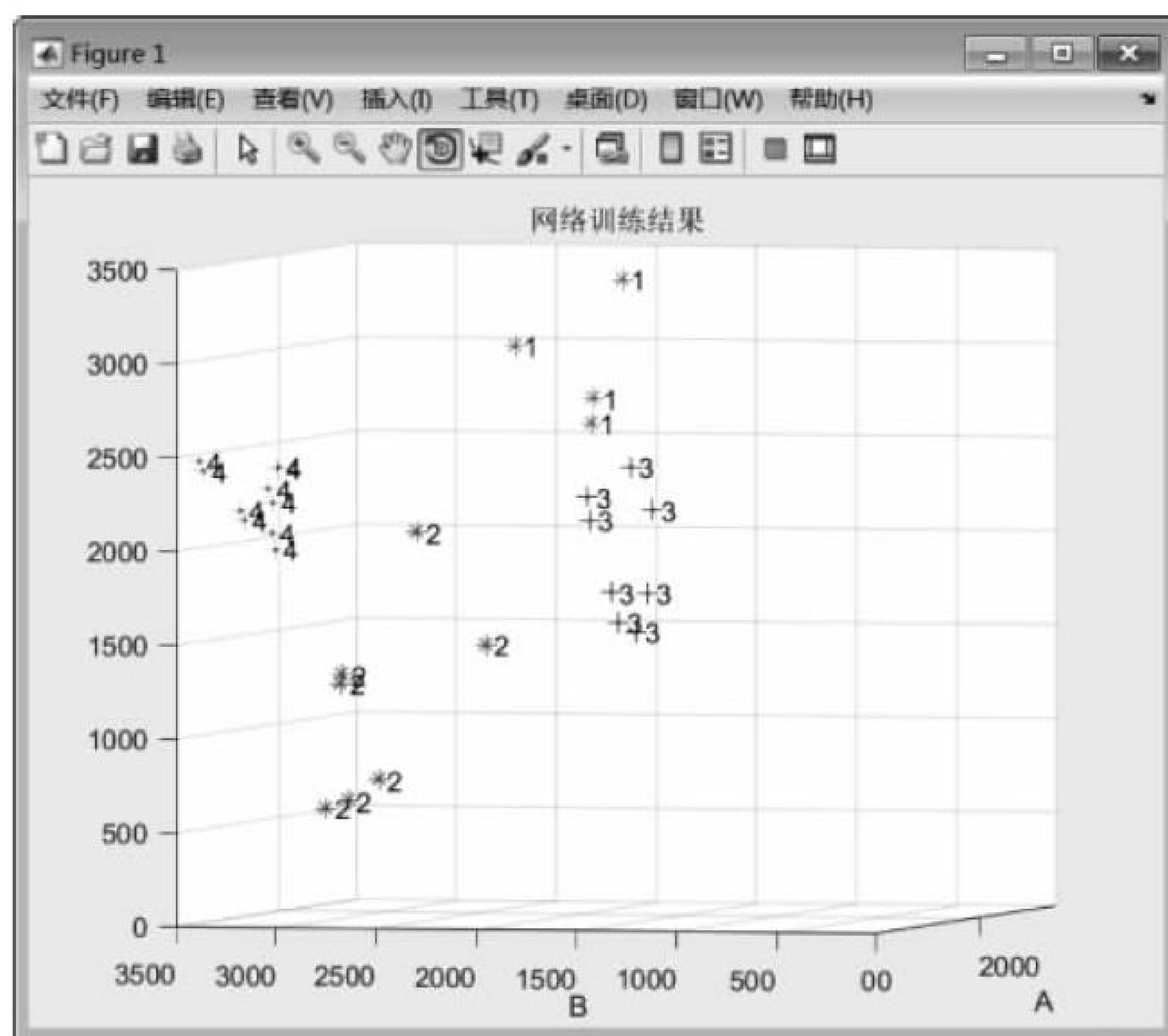


图 6-45　训练结果类别分类图

训练后的 GRNN 对训练数据进行分类后的结果与目标结果对比如表 6-6 所示。

表 6-6　训练后的 GRNN 对训练数据进行分类后的结果与目标结果对比

序　号	A	B	C	目标结果	GRNN 网络分类结果
4	864.45	1647.31	2665.9	1	1
6	877.88	2031.66	3071.18	1	1
16	1418.79	1775.89	2772.9	1	1
25	1449.58	1641.58	3405.12	1	1
8	2352.12	2557.04	1411.53	2	2
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
22	2802.88	3017.11	1984.98	2	2
24	2063.54	3199.76	1257.21	2	2
1	1739.94	1675.15	2395.96	3	3
3	1756.77	1652	1514.98	3	3
7	1803.58	1583.12	2163.05	3	3
11	1571.17	1731.04	1735.33	3	3
17	1845.59	1918.81	2226.49	3	3
20	1692.62	1867.5	2108.97	3	3
21	1680.67	1575.78	1725.1	3	3
26	1651.52	1713.28	1570.38	3	3
2	373.3	3087.05	2429.47	4	4
5	222.85	3059.54	2002.33	4	4
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
23	172.78	3084.49	2328.65	4	4
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4

训练后的 GRNN 对训练数据进行分类后的结果与目标结果完全吻合，可见 GRNN 训练效果良好。

继续执行程序，可得到待分类样本的分类结果：

```
ac =
1 ～ 15 列
3    3    1    3    4    2    2    3    4    1    3    3    1    2
4
16 ～ 30 列
2    4    3    4    2    2    3    3    1    1    4    1    3    3
3
```

6.6.5 结论

从分类结果可以看出广义神经网络的分类效果优于径向基神经网络，广义神经网络能够以任意精度逼近任意非线性连续函数。其网络训练非常简单，当训练样本通过隐含层的同时，网络训练随即完成。它的训练过程不需要迭代，它比 BP 网络的训练过程快得多，更适合于在线数据的实时处理。

由于简单的网络结构，我们不需要对网络的隐含层数和隐含单元的个数进行估算和猜测。由于它是从径向基函数引申而来，因此只有一个自由参数，即径向基函数的平滑参数。而它的优化值通过交叉验证的方法很容易得到。

6.7 小波神经网络

小波分析是 20 世纪 80 年代中期发展起来的一门新的数学理论和方法，是时间—频率分析领域的一种新技术。小波分析的基本思想类似于傅里叶变换，其函数通过一族基函数在空间上的投影来表征。神经网络起源于 20 世纪 40 年代，是由大量简单的处理单元(神经元)广泛地互相连接形成的复杂网络系统，它反映了人脑功能的许多基本特征，是一个高度复杂的非线性动力学系统。由于小波变换能够反映信号的时频局部特性和聚焦特性，而神经网络在信号处理方面具有自学习、自适应、健壮性、容错性等能力，如何把二者的优势结合起来一直是人们所关心的问题。小波神经网络正是小波分析和神经网络相结合的产物。

6.7.1 小波神经网络的基本结构

小波变换被认为是傅里叶发展史上一个新的里程碑，它克服了傅里叶分析不能作局部分析的缺点，是傅里叶分析划时代发展的结果。随着小波理论发展日益成熟，其应用领域也变得十分广泛，特别是在信号处理、数值计算、模式识别、图像处理、语音分析、量子物理、生物医学工程、计算机视觉、故障诊断及众多非线性领域等，小波变换都在不断发展之中。

神经网络是在现代神经学的研究基础上发展起来的一种模仿人脑信息处理机制的网络系统，它具有自组织、自学习和极强的非线性处理能力，能够完成学习、记忆、识别和推理等功能。神经网络的崛起，对认知和智力本质的基础研究乃至计算机产业都产生了空前的刺激和极大的推动作用。

目前，小波分析与神经网络主要有两种结合方式：一种是“松散型”，如图 6-46 所示，即先用小波分析对信号进行预处理，然后再送入神经网络处理；另一种是“紧致型”，如图 6-47 所示，即小波神经网络(wavelet neural network)或小波网络，它是结合小波变换理论与神经网络的思想而构造的一种新的神经网络模型。其方法是将神经网络隐含层中神经元的传递激发函数用小波函数来代替，充分继承了小波变换良好的时频局部化性质及神经网络的自学习功能的特点，被广泛运用于信号处理、数据压缩、模式识别和故障诊断等领域。“紧致型”小波神经网络具有更好的数据处理能力，是小波神经网络的研究方向。在图 6-47 中，有

输入层、隐含层和输出层，输出层采用线性输出，输入层有 $m(m=1,2,\cdots,M)$ 个神经元，隐含层有 $k(k=1,2,\cdots,K)$ 个神经元，输出层有 $n(n=1,2,\cdots,N)$ 个神经元。

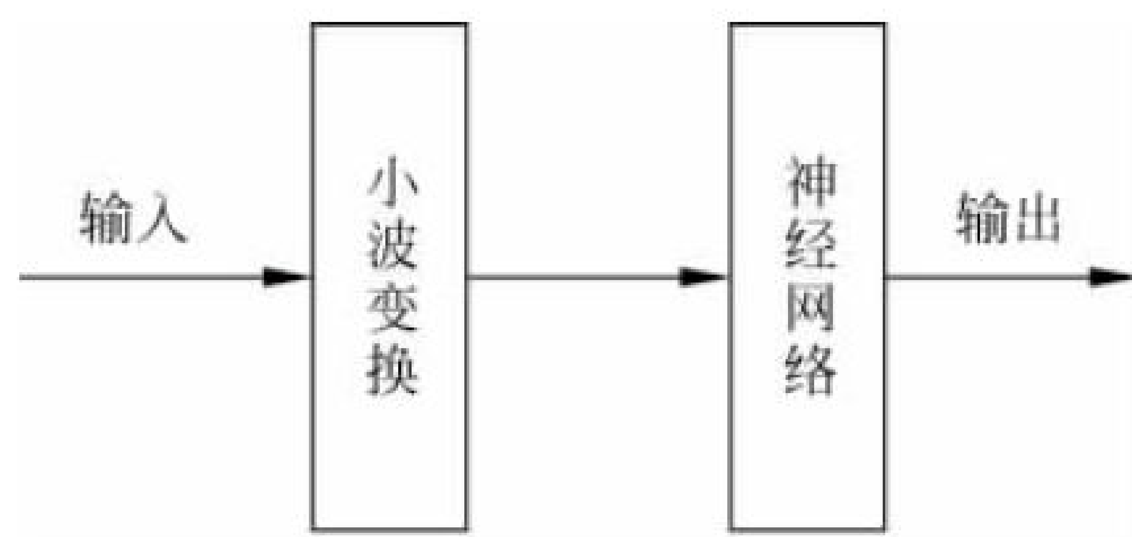


图 6-46　小波神经网络松散型结构

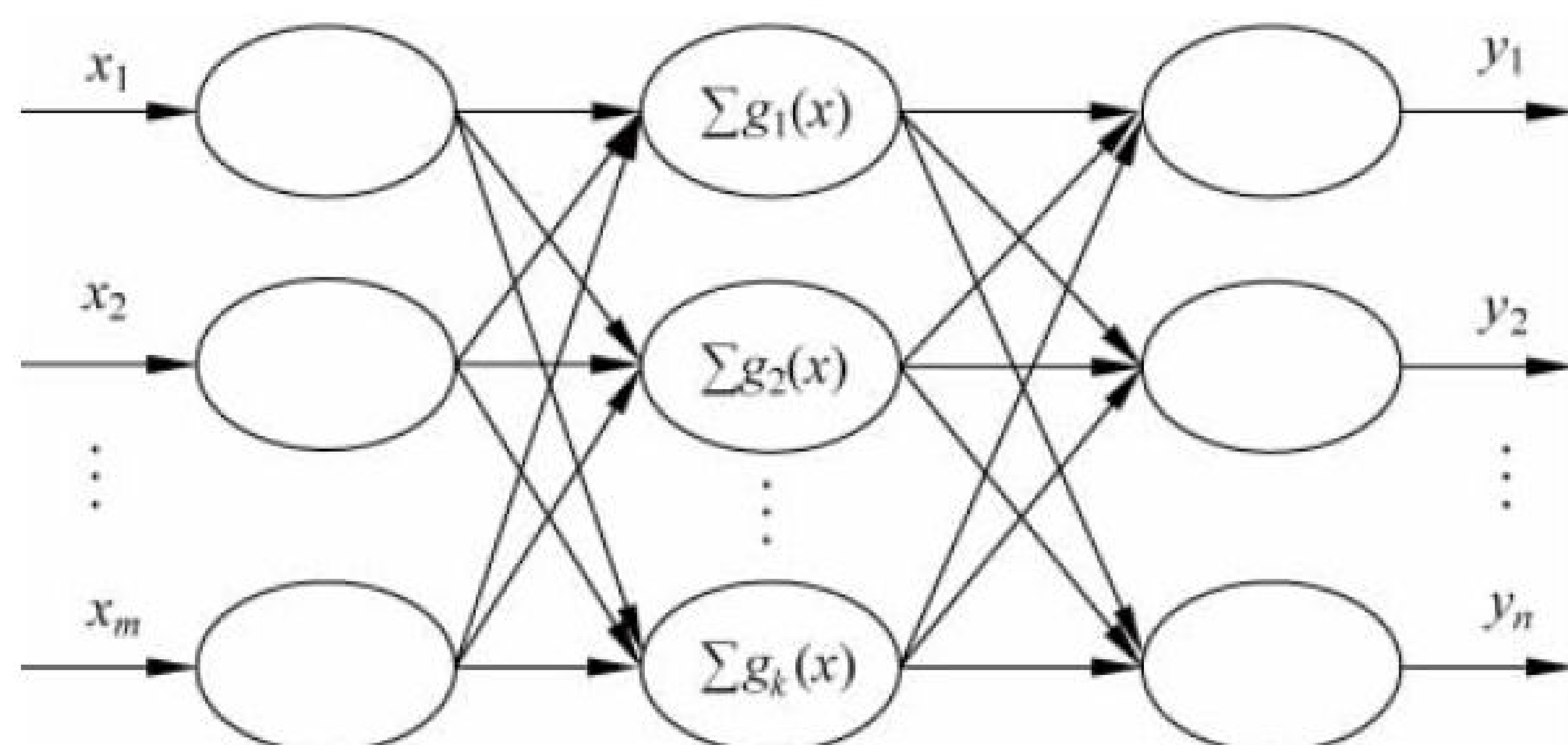


图 6-47　小波神经网络紧致型结构

根据基函数 $g_k(x)$ 和学习参数的不同，图 6-47 中小波神经网络的结果可分为 3 类。

(1) 连续参数的小波神经网络。这是小波最初被提出采用的一种形式。令图 6-47 中基函数为

$$g_j(x)=\prod_{i=1}^{M}\Psi\left(\frac{x_i-b_{ij}}{a_{ij}}\right)=\Psi(A_jx-b'_j) \tag{6-64}$$

则网络输出为

$$\hat{y}_i=\sum_{j=1}^{K}C_{ji}g_j(x) \tag{6-65}$$

其中 $1\leqslant j\leqslant K$，$A_j=\mathrm{diag}(a_{1j}^{-1},L,a_{mj}^{-1})$，$1\leqslant i\leqslant N$，$b'_j=[a_{1j}^{-1}b_{ij},L,a_{mj}^{-1}b_{mj}]^{\mathrm{T}}$，在网络学习中尺度因子 a_{ij}、平移因子 b_{ij}、输出权值 C_{ij} 一起通过某种修正。这种小波网络类似于径向基函数网络，借助于小波分析理论，可使网络具有较简单的拓扑结构和较快的收敛速度。但由于尺度和评议参数均可调，使其与输出为非线性关系，通常需利用非线性优化方法进行参数修正，易带来类似 BP 网络参数修正时存在局部极小值的弱点。

(2) 由框架作为基函数的小波神经网络。由于不考虑正交性，小波函数的选取有很大的自由度。令图 6-47 中的基函数为

$$g(x)=\prod_{i=1}^{P}g(x_i)=\prod_{i=1}^{P}\Psi(2_jx_i-k) \tag{6-66}$$

则网络输出为

$$\hat{y}_i=\sum C_{j,k}\Psi_{j,k} \tag{6-67}$$

根据函数 f 的时频特性确定取值范围后，网络的可调参数只有权值，其与输出呈线性关系，可通过最小二乘法或其他优化法修正权值，使网络能充分逼近 $f(x)$。

这种形式的网络虽然基函数选取灵活，但由于框架可以是线性相关的，使得网络函数的个数有可能存在冗余，对过于庞大的网络需考虑优化结构算法。

(3) 基于多分辨分析的正交基小波网络。网络隐节点由小波节点 Ψ 和尺度函数节点 φ 构成，网络输出为

$$\hat{y}_i(x)=\sum_{j=L,k=z} d_{j,k}\varphi_{j,k}(x)+\sum_{j\geqslant L,k\in x} C_{j,k}\Psi_{j,k}(x) \tag{6-68}$$

当尺度 L 足够大时，忽略式(6-68)右端第 2 项表示的小波细节分量，这种形式的小波网络的主要依据是 Daubechies 的紧支撑正交小波及 Mallat 的多分辨分析理论。

尽管正交小波网络在理论上研究较为方便，但正交基函数的构造复杂，不如一般的基于框架的小波网络实用。

6.7.2 小波神经网络的训练算法

小波神经网络最早是由法国著名的信息科学机构 IRISA 的 Zhang Qinghua 等于 1992 年提出的，是在小波分析的基础上提出的一种多层前馈模型网络，可以使网络从根本上避免局部最优并且加快了收敛速度，具有很强的学习和泛化能力。小波神经网络是用非线性小波基取代通常的非线性 sigmoid 函数，其信号表述是通过将所选取的小波基进行线性叠加来表现的。

设小波神经网络有 m 个输入节点、N 个输出节点、n 个隐层节点。网络的输入和输出数据分别用向量 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 来表示，即

$$\boldsymbol{X}=(x_1,x_2,\cdots,x_m),\quad \boldsymbol{Y}=(y_1,y_2,\cdots,y_n)$$

若设 x_k 为输入层的第 k 个输入样本，y_i 为输出层的第 i 个输出值，w_{ij} 为连接输出层节点 i 和隐含层节点 j 的权值，w_{jk} 为连接隐含层节点 j 和输出层节点 k 的权值。令 w_{i0} 是第 j 个输出层节点阈值，w_{j0} 是第 j 个隐含层节点阈值(相应的输入 $x_0=-1$)，a_j 为第 j 个隐含层节点的伸缩因子、b_j 为第 j 个隐含层节点的平移因子，则小波神经网络模型为

$$\boldsymbol{y}_i=\sigma\left[\sum_{i=0}^{n}\omega_{ij}\psi_{a,b}\left(\sum_{k=0}^{m}\omega_{jk}x_k(t)\right)\right] \tag{6-69}$$

式中，$i=1,2,\cdots,N$；$\sigma(t)=\dfrac{1}{1+\mathrm{e}^{-t}}$。令 $net_j=\sum_{k=0}^{m}\omega_{jk}x_k$，且

$$\psi_{a,b}(net_j)=\frac{(net_j-b_j)}{a_j} \tag{6-70}$$

则
$$\boldsymbol{y}_i=\sigma\left[\sum_{i=0}^{n}\omega_{ij}\psi_{a,b}(net_j)\right] \tag{6-71}$$

给定样本集 $\{(x_i,y_i)\}$，$i=1,2,\cdots,N$ 后，网络的权值被调整，使如下的误差目标函数达到最小

$$E(\boldsymbol{w})=\frac{1}{2}\sum_{i=1}^{n}\|Y_i-\boldsymbol{d}_i\|^2 \tag{6-72}$$

式中，d_i 为网络的输出向量，$\boldsymbol{W}$ 为网络中所有权值组成的权向量，$\boldsymbol{W}\in R^t$。网络的学习可以归结为如下的无约束最优化问题：

小波神经网络采用梯度法，即最快下降法来求解该问题，那么小波网络的权值的调整规

则处理过程分为两个阶段：一是从网络的输入层开始逐层向前计算，根据输入样本计算各层的输出，最终求出网络输出层的输出，这是前向传播过程；二是对权值的修正，从网络的输出层开始逐层向后进行计算和修正，这是反向传播过程。两个过程反复交替，直到收敛为止。通过不断修正权值 $\boldsymbol{W}$，使 $E(\bullet)$ 达到最小值。

令 d_i^p 为第 P 个模式第 i 个期望输出，基于最小二乘的代价函数可表示为

$$E=\frac{1}{2}\sum_{p=1}^{p}\sum_{i=1}^{N}(\boldsymbol{d}_i^p-y_i^p)^2 \tag{6-73}$$

则可以计算得到下列偏导数

$$\frac{\partial E}{\partial \omega_{ij}}=-\sum_{p=1}^{p}(\boldsymbol{d}_i^p-y_i^p)y_i^p(1-y_i^p)\psi_{a,b}(net_j^p) \tag{6-74}$$

$$\frac{\partial E}{\partial \omega_{jk}}=-\sum_{p=1}^{p}\sum_{i=1}^{N}(\boldsymbol{d}_i^p-y_i^p)y_i^p(1-y_i^p)\omega_{ij}\psi'_{a,b}(net_j^p)\boldsymbol{x}_k^p/a_j \tag{6-75}$$

$$\frac{\partial E}{\partial a_j}=-\sum_{p=1}^{p}\sum_{i=1}^{N}(\boldsymbol{d}_i^p-y_i^p)y_i^p(1-y_i^p)\omega_{ij}\psi'_{a,b}(net_j^p)\frac{(net_j^p-b_j)}{a_j}/a_j \tag{6-76}$$

$$\frac{\partial E}{\partial b_j}=-\sum_{p=1}^{p}\sum_{i=1}^{N}(\boldsymbol{d}_i^p-y_i^p)y_i^p(1-y_i^p)\omega_{ij}\psi'_{a,b}(net_j^p)/a_j \tag{6-77}$$

为了加快算法的收敛速度，引入动量因子 α，因此权向量的迭代公式为

$$\omega_{ij}(t+1)=\omega_{ij}(t)-\eta\frac{\partial E}{\partial \omega_{ij}}+\alpha\Delta\omega_{ij}(t) \tag{6-78}$$

$$\omega_{jk}(t+1)=\omega_{jk}(t)-\eta\frac{\partial E}{\partial \omega_{jk}}+\alpha\Delta\omega_{jk}(t) \tag{6-79}$$

$$a_j(t+1)=a_j(t)-\eta\frac{\partial E}{\partial a_j}+\alpha\Delta a_j(t) \tag{6-80}$$

$$b_j(t+1)=b_j(t)-\eta\frac{\partial E}{\partial b_j}+\alpha\Delta b_j(t) \tag{6-81}$$

在网络权值的调整过程中，往往是在学习的初始阶段，学习步长选择大一些，以使学习速度加快；当接近最佳点时，学习速率选择小一些，否则连接权值将产生振荡而难以收敛。学习步长调整的一般规则是：在连续迭代几步过程中，若新误差大于旧误差，则学习速率减小；若新误差小于旧误差，则学习步长增大。

6.7.3 小波神经网络结构设计

1. 小波函数的选择

小波的选择具有相对的灵活性，对不同的数据信号，x 要选择恰当的小波作为分解基。小波变换不像傅里叶变换那样是由正弦函数唯一决定的，小波基可以有很多种，不同的小波适合不同的信号。

(1) Mexican hat 和 Morlet 小波基没有尺度函数，是非正交小波基。其优点是函数对称且表达式清楚简单，缺点是无法对分解后的信号进行重构。采用 Morlet 小波(r 通常取值为 1.75)构造的小波网络已经被用于各种领域。

(2) Daubechies 是一种具有紧支撑的正交小波，随着 N 的增加，Daubechies 小波的时域支撑长度变长；矩阵阶数增加；特征正则性增加，幅频特性也越接近理想。当选取 N 值越大的高阶 db 小波时，其构成可近似看成是一个理想的低通滤波器和理想的带通滤波器，且具有能量无损性。

通常在信号的近似和估计作用中，小波函数的选择应与信号的特征匹配，应考虑小波的波形、支撑大小和消失矩阵的数目。连续的小波基函数都在有效支撑区域之外快速衰减。有效支撑区域越长，频率分辨率越好；有效支撑区域越短，时间分辨率越好。如果进行时频分析，则要选择光滑的连续小波，因为时域越光滑的基函数在频域的局部化特性方面越好。如果进行信号检测，则应尽量选择与信号波形相近似的小波。

2. 隐含层节点的选取

隐含层节点的作用是从样本中提取并存储其内在规律，每个隐含层节点有若干个权值，而每个权值都是增强网络映射能力的参数。隐含层节点数量太少，网络从样本中获取信息的能力就差，不足以概括和体现训练集中的样本规律；隐含层节点数目太多，又可能把样本中非规律性的内容记牢，从而出现所谓的“过拟合”问题，反而降低了网络的泛化能力。此外，隐含层节点数过多会增加神经网络的训练时间。

6.7.4 小波神经网络分类器的 MATLAB 实现

1. 程序模块介绍

程序中用到的小波神经网络工具箱调用代码如下：

```
function y = d_mymorlet(t)
y =  - 1.75 * sin(1.75 * t). * exp( - (t.^2)/2) - t *  cos(1.75 * t). * exp( - (t.^2)/2);
function y = mymorlet(t)
y =  exp( - (t.^2)/2)  *  cos(1.75 * t);
```

1) 初始化程序

在程序开始，首先需要设置网络的相关参数。

网络参数配置的 MATLAB 程序代码如下：

```
load wavelet2 input output input_test output_test
M = size(input,2);            % 输入节点个数
N = size(output,2);           % 输出节点个数
n = 10;                       % 隐形节点个数
lr1 = 0.01;                   % 学习概率
lr2 = 0.001;                  % 学习概率
maxgen = 200;                 % 迭代次数
```

权值初始化代码如下：

```
Wjk = randn(n,M);Wjk_1 = Wjk;Wjk_2 = Wjk_1;              %% Wjk 和 Wij 为网络链接权重
Wij = randn(N,n);Wij_1 = Wij;Wij_2 = Wij_1;
a = randn(1,n);a_1 = a;a_2 = a_1;                        %% 小波函数伸缩因子
b = randn(1,n);b_1 = b;b_2 = b_1;                        %% 小波函数平移因子
```

节点初始化代码如下：

```
y = zeros(1,N);
net = zeros(1,n);
net_ab = zeros(1,n);
```

权值学习增量初始化代码如下：

```
d_Wjk = zeros(n,M);
d_Wij = zeros(N,n);
d_a = zeros(1,n);
d_b = zeros(1,n);
```

输入输出数据归一化处理代码如下：

```
[inputn,inputps] = mapminmax(input');
[outputn,outputps] = mapminmax(output');
inputn = inputn';
outputn = outputn';
```

2）网络训练程序

网络训练的 MATLAB 程序代码如下：

```
for kk = 1:size(input,1)
          x = inputn(kk,:);
          yqw = outputn(kk,:);
          for j = 1:n
              for k = 1:M
                  net(j) = net(j) + Wjk(j,k) * x(k);
                  net_ab(j) = (net(j) - b(j))/a(j);
              end
              temp = mymorlet(net_ab(j));
              for k = 1:N
                  y = y + Wij(k,j) * temp;          % 小波函数
              end
          end
```

3）计算误差和

计算误差和使用如下语句：

```
error(i) = error(i) + sum(abs(yqw - y));
```

4）权值调整

权值调整的程序代码如下：

```
        for j = 1:n
            %计算 d_Wij
            temp = mymorlet(net_ab(j));
            for k = 1:N
                d_Wij(k,j) = d_Wij(k,j) - (yqw(k) - y(k)) * temp;
            end
            %计算 d_Wjk
            temp = d_mymorlet(net_ab(j));
            for k = 1:M
                for l = 1:N
                    d_Wjk(j,k) = d_Wjk(j,k) + (yqw(l) - y(l)) * Wij(l,j);
                end
                d_Wjk(j,k) = - d_Wjk(j,k) * temp * x(k)/a(j);
            end
            %计算 d_b
            for k = 1:N
                d_b(j) = d_b(j) + (yqw(k) - y(k)) * Wij(k,j);
            end
            d_b(j) = d_b(j) * temp/a(j);
            %计算 d_a
            for k = 1:N
                d_a(j) = d_a(j) + (yqw(k) - y(k)) * Wij(k,j);
            end
            d_a(j) = d_a(j) * temp * ((net(j) - b(j))/b(j))/a(j);
        end
        %权值参数更新
        Wij = Wij - lr1 * d_Wij;
        Wjk = Wjk - lr1 * d_Wjk;
        b = b - lr2 * d_b;
        a = a - lr2 * d_a;
        d_Wjk = zeros(n,M);
        d_Wij = zeros(N,n);
        d_a = zeros(1,n);
        d_b = zeros(1,n);
        y = zeros(1,N);
        net = zeros(1,n);
        net_ab = zeros(1,n);
        Wjk_1 = Wjk;Wjk_2 = Wjk_1;
        Wij_1 = Wij;Wij_2 = Wij_1;
        a_1 = a;a_2 = a_1;
        b_1 = b;b_2 = b_1;
    end
end
```

5）网络预测

网络预测的程序代码如下：

```
for i = 1:10
    x_test = x(i,:);
```

```
        for j = 1:1:n
            for k = 1:1:M
                net(j) = net(j) + Wjk(j,k) * x_test(k);
                net_ab(j) = (net(j) - b(j))/a(j);
            end
            temp = mymorlet(net_ab(j));
            for k = 1:N
                y(k) = y(k) + Wij(k,j) * temp;
            end
        end
        yuce(i) = y(k);
        y = zeros(1,N);
        net = zeros(1,n);
        net_ab = zeros(1,n);
    end
```

2. MATLAB完整程序及仿真结果

小波神经网络数据分类的完整MATLAB程序代码如下：

```
%% 清空环境变量
clc
clear
%% 网络参数配置
load wavelet2 input output input_test output_test
M = size(input,2);                                  %输入节点个数
N = size(output,2);                                 %输出节点个数
n = 10;                                             %隐形节点个数
lr1 = 0.01;                                         %学习概率
lr2 = 0.001;                                        %学习概率
maxgen = 200;                                       %迭代次数
%权值初始化
Wjk = randn(n,M);Wjk_1 = Wjk;Wjk_2 = Wjk_1;         %% Wjk 和 Wij 为网络连接权重
Wij = randn(N,n);Wij_1 = Wij;Wij_2 = Wij_1;
a = randn(1,n);a_1 = a;a_2 = a_1;                   %% 小波函数伸缩因子
b = randn(1,n);b_1 = b;b_2 = b_1;                   %% 小波函数平移因子
%节点初始化
y = zeros(1,N);
net = zeros(1,n);
net_ab = zeros(1,n);
%权值学习增量初始化
d_Wjk = zeros(n,M);
d_Wij = zeros(N,n);
d_a = zeros(1,n);
d_b = zeros(1,n);
%% 输入输出数据归一化
[inputn,inputps] = mapminmax(input');
[outputn,outputps] = mapminmax(output');
inputn = inputn';
outputn = outputn';
%% 网络训练
for i = 1:maxgen
```

```
%误差累计
error(i) = 0;
% 循环训练
for kk = 1:size(input,1)
    x = inputn(kk,:);
    yqw = outputn(kk,:);
    for j = 1:n
        for k = 1:M
            net(j) = net(j) + Wjk(j,k) * x(k);
            net_ab(j) = (net(j) - b(j))/a(j);
        end
        temp = mymorlet(net_ab(j));
        for k = 1:N
            y = y + Wij(k,j) * temp; %小波函数
        end
    end
    %计算误差和
    error(i) = error(i) + sum(abs(yqw - y));
    %权值调整
    for j = 1:n
        %计算 d_Wij
        temp = mymorlet(net_ab(j));
        for k = 1:N
            d_Wij(k,j) = d_Wij(k,j) - (yqw(k) - y(k)) * temp;
        end
        %计算 d_Wjk
        temp = d_mymorlet(net_ab(j));
        for k = 1:M
            for l = 1:N
                d_Wjk(j,k) = d_Wjk(j,k) + (yqw(l) - y(l)) * Wij(l,j);
            end
            d_Wjk(j,k) = - d_Wjk(j,k) * temp * x(k)/a(j);
        end
        %计算 d_b
        for k = 1:N
            d_b(j) = d_b(j) + (yqw(k) - y(k)) * Wij(k,j);
        end
        d_b(j) = d_b(j) * temp/a(j);
        %计算 d_a
        for k = 1:N
            d_a(j) = d_a(j) + (yqw(k) - y(k)) * Wij(k,j);
        end
        d_a(j) = d_a(j) * temp * ((net(j) - b(j))/b(j))/a(j);
    end
    %权值参数更新
    Wij = Wij - lr1 * d_Wij;
    Wjk = Wjk - lr1 * d_Wjk;
    b = b - lr2 * d_b;
    a = a - lr2 * d_a;
    d_Wjk = zeros(n,M);
    d_Wij = zeros(N,n);
    d_a = zeros(1,n);
    d_b = zeros(1,n);
```

```
        y = zeros(1,N);
        net = zeros(1,n);
        net_ab = zeros(1,n);
        Wjk_1 = Wjk;Wjk_2 = Wjk_1;
        Wij_1 = Wij;Wij_2 = Wij_1;
        a_1 = a;a_2 = a_1;
        b_1 = b;b_2 = b_1;
    end
end
%% 网络预测
%预测输入归一化
x = mapminmax('apply',input_test',inputps);
x = x';
%网络预测
for i = 1:10
    x_test = x(i,:);
    for j = 1:1:n
        for k = 1:1:M
            net(j) = net(j) + Wjk(j,k) * x_test(k);
            net_ab(j) = (net(j) - b(j))/a(j);
        end
        temp = mymorlet(net_ab(j));
        for k = 1:N
            y(k) = y(k) + Wij(k,j) * temp;
        end
    end
    yuce(i) = y(k);
    y = zeros(1,N);
    net = zeros(1,n);
    net_ab = zeros(1,n);
end
%预测输出反归一化
ynn = mapminmax('reverse',yuce,outputps);
ynnn = roundn(ynn,0);
if ynnn >= 4
    yn = 4
elseif ynnn >= 1
    yn = ynnn
else yn = 1
end;
%% 结果分析
figure(1)
plot(yn,'r * :')
hold on
plot(output_test,'bo -- ')
title('预测分类','fontsize',12)
legend('预测分类','实际分类')
xlabel('数据组')
ylabel('类别')
%% 误差显示
figure(2)
plot(error,'g')
title('网络进化过程','fontsize',12)
xlabel('进化次数')
ylabel('预测误差')
```

(1) 将前 29 组数据(29×3)作为训练输入,后 30 组数据(30×3)作为测试输入,并将得到的结果与实际分类进行比对。

设置迭代次数为 200 次,网络进化过程如图 6-48 所示,预测分类结果如图 6-49 所示。

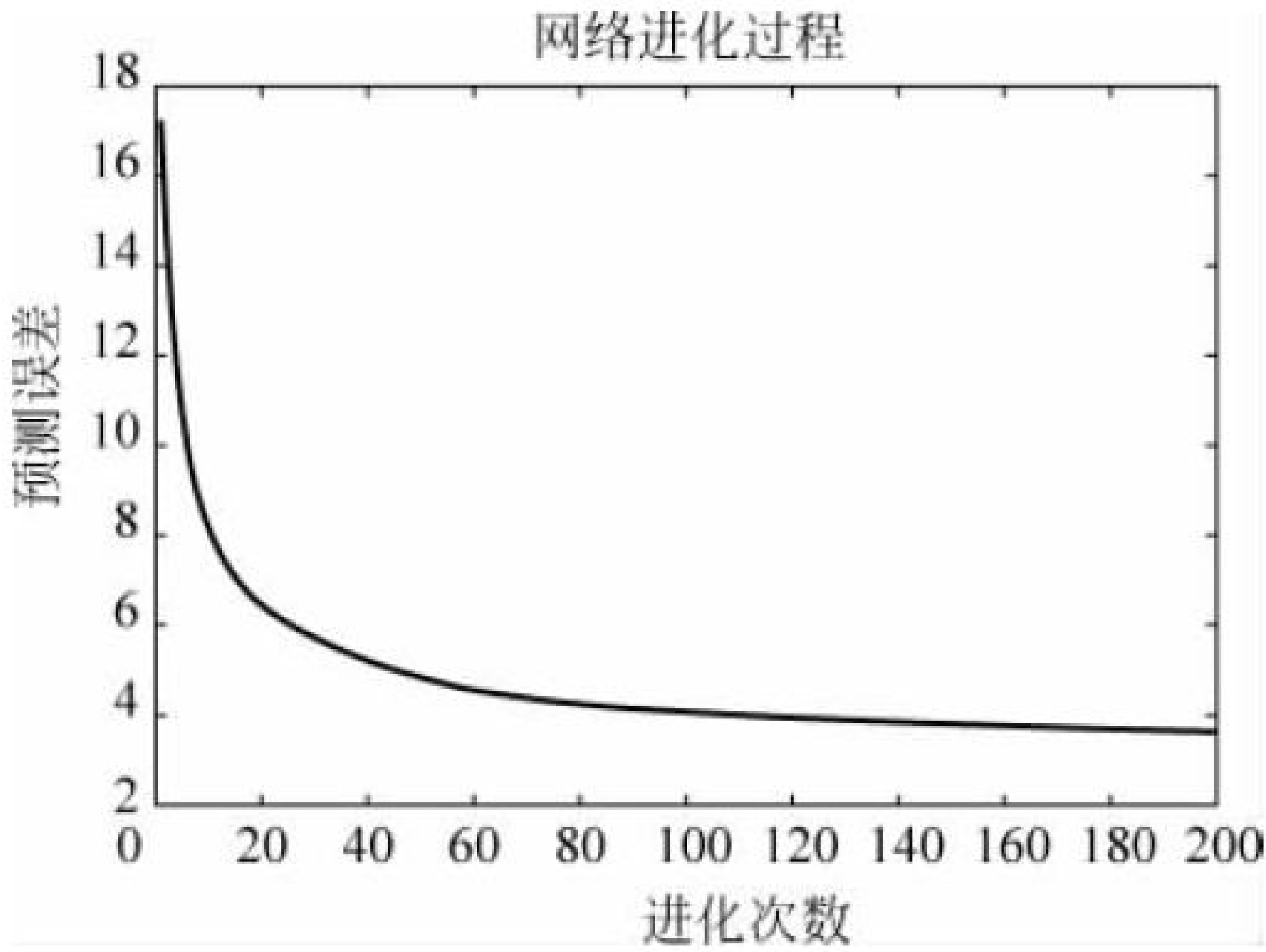


图 6-48　网络进化过程(200 次)

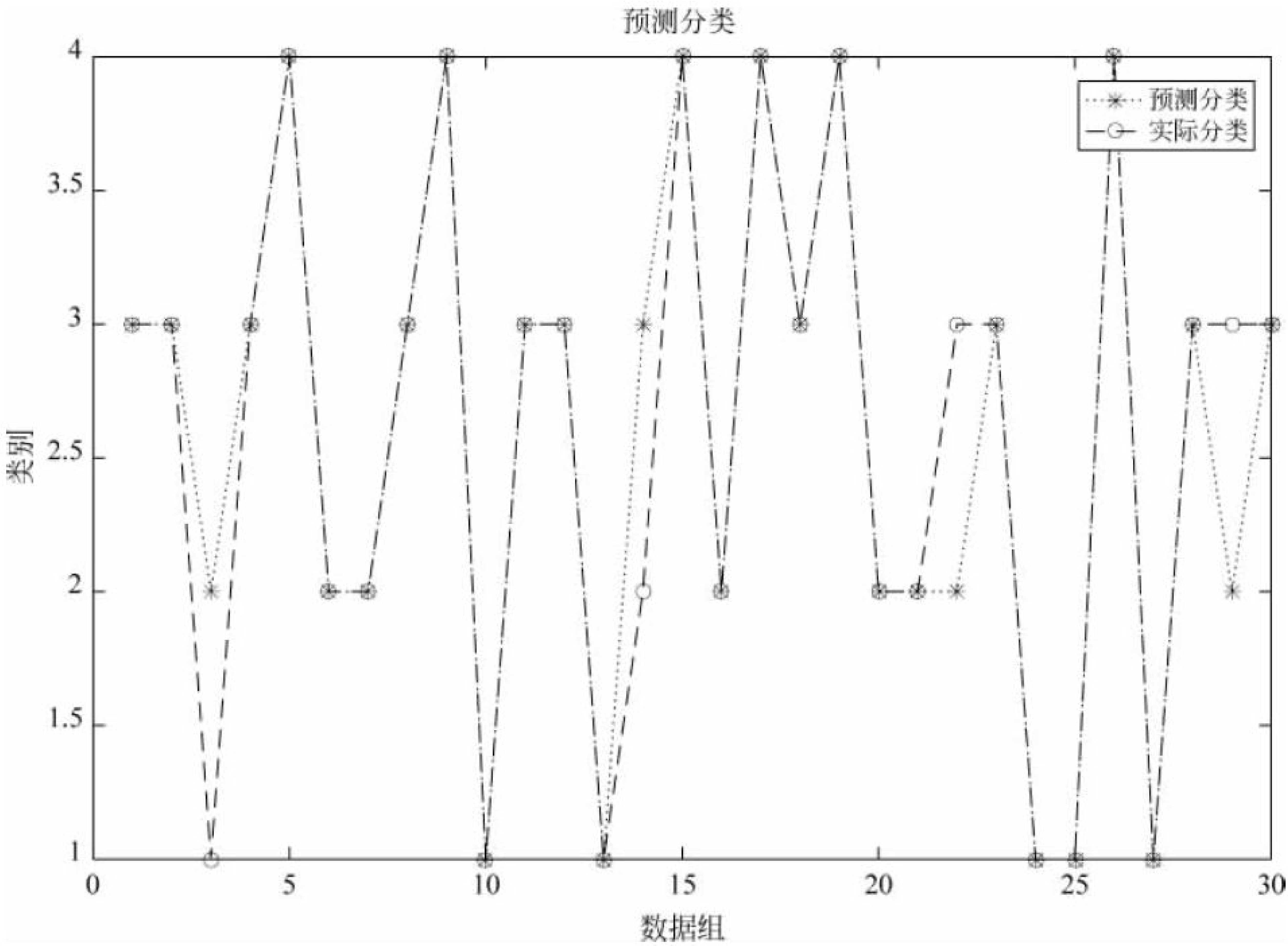


图 6-49　预测分类结果(200 次)

对预测分类的输出结果(yn)如下:

```
yn =
1 ～ 15 列
     3     3     2     3     4     2     2     3     4     1     3     3     1     3     4
16 ～ 30 列
     2     4     3     4     2     2     2     3     1     1     4     1     3     2     3
```

由网络的进化过程曲线可以看出,当进化次数为100次的时候,误差已经趋于稳定,学习速率较快,但是由预测分类的结果可以看出,30组数据的分类结果并没有和实际分类完全重合,有4组数据的分类不准确,错误率约为4/30(约13%)。因此尝试增加迭代次数,进行第二次分类。

设置迭代次数为300次,网络进化过程如图6-50所示,预测分类结果如图6-51所示。

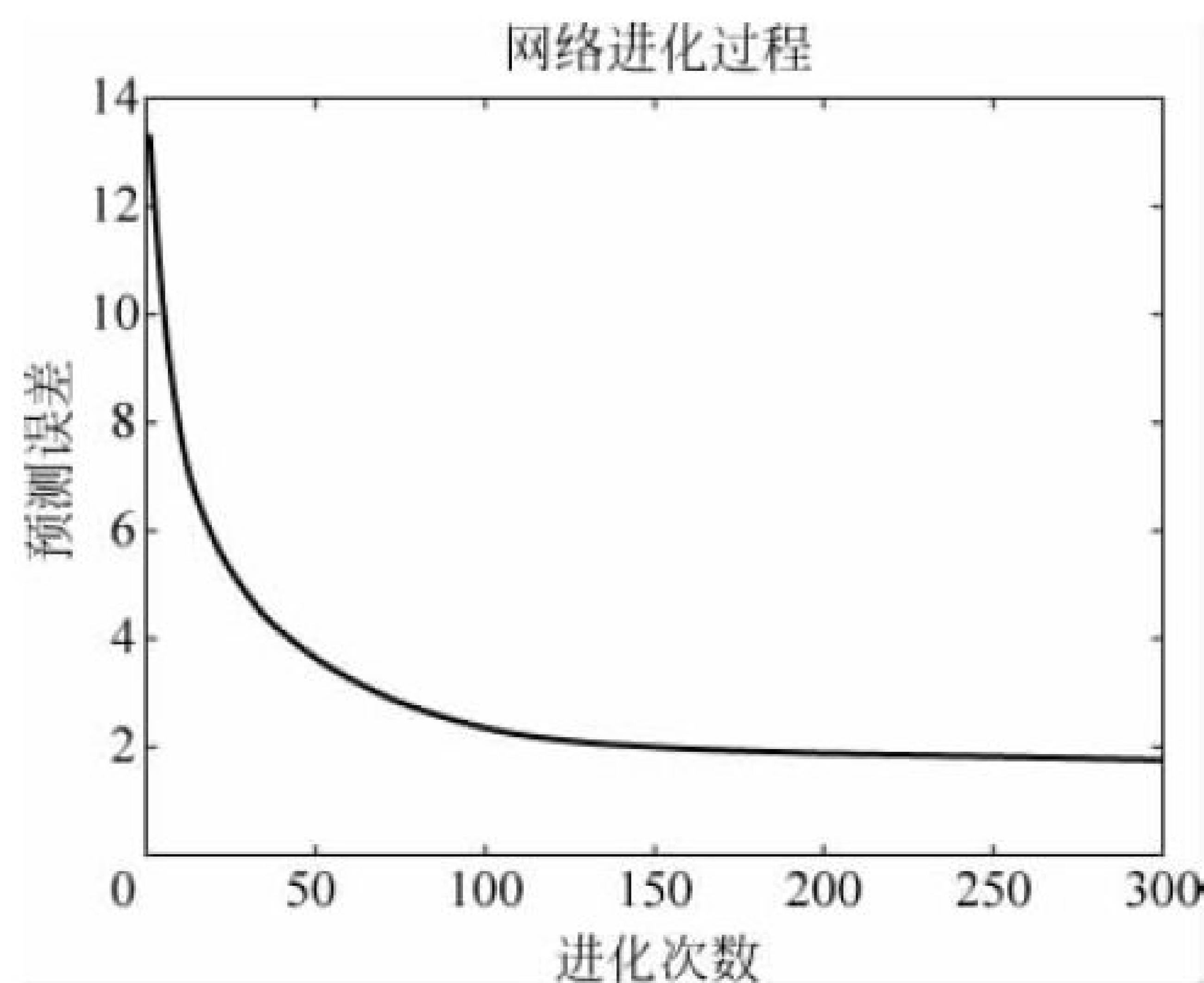


图6-50 网络进化过程(300次)

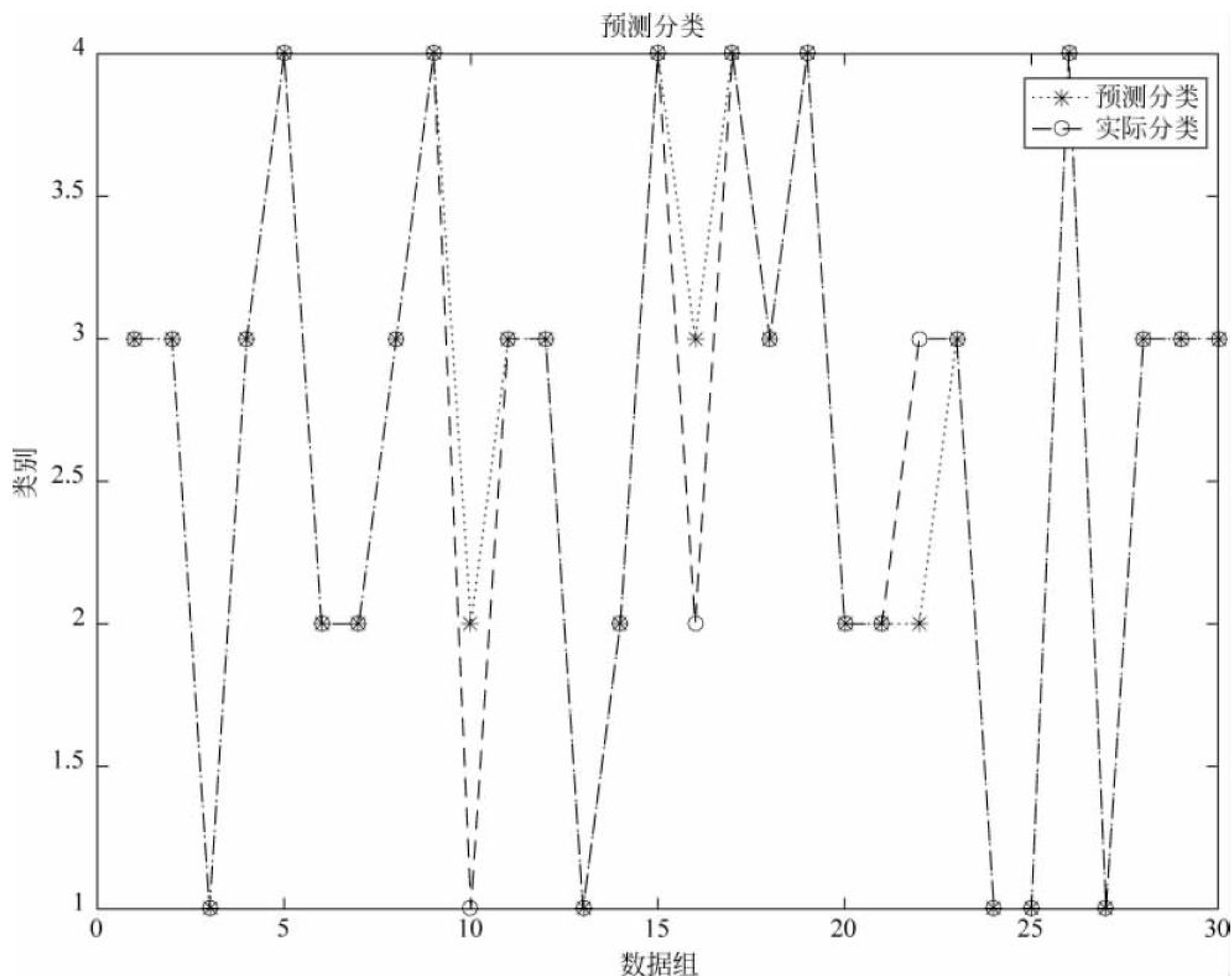


图6-51 预测分类结果(300次)

对预测分类的输出结果(yn)如下:

```
yn =
1 ～ 15 列
3    3    1    3    4    2    2    3    4    2    3    3    1    2    4
16 ～ 30 列
3    4    3    4    2    2    2    3    1    1    4    1    3    3    3
```

由网络的进化过程曲线可以看出，当进化次数为 100 次时，误差已经趋于稳定，学习速率较快，但是由预测分类的结果可以看出，30 组数据的分类结果依然没有和实际分类完全重合，有 3 组数据的分类是不准确的，错误率约为 3/30(10%)，亦即通过增加迭代次数，错误率有一定的下降。下面将尝试继续增加训练次数，进行第三次分类。

设置迭代次数为 1000 次，网络进化过程如图 6-52 所示，预测分类结果如图 6-53 所示。

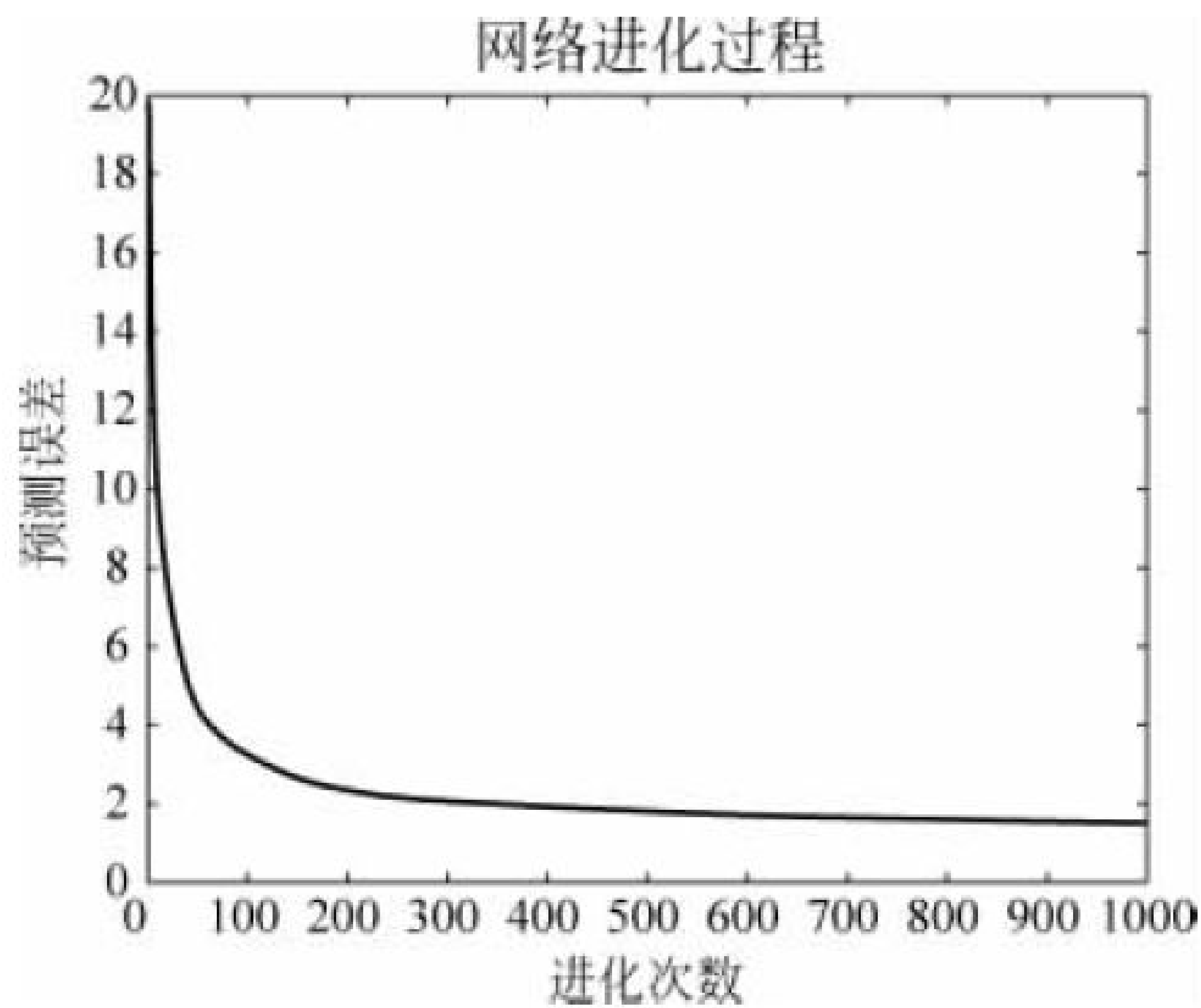


图 6-52　网络进化过程(1000 次)

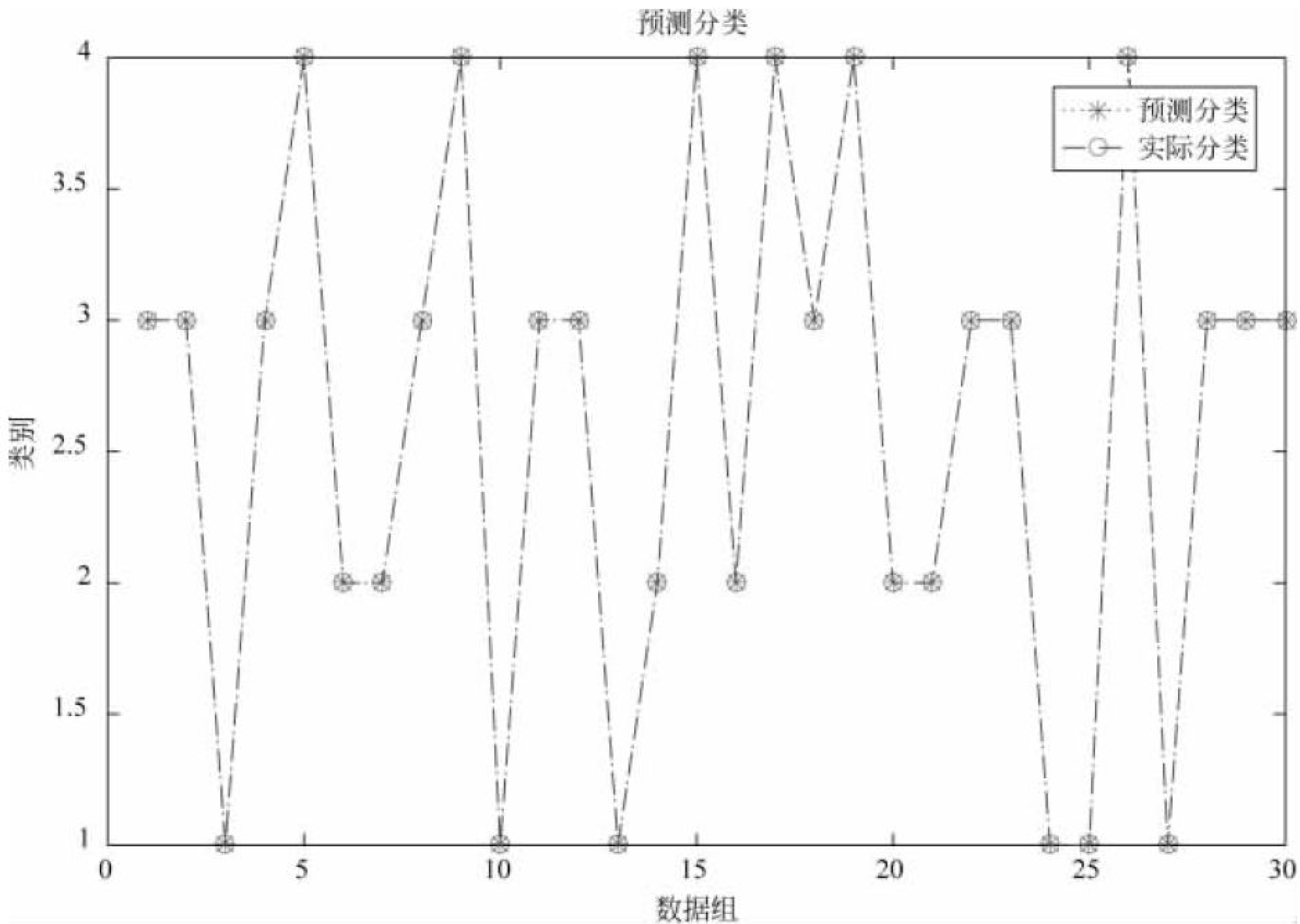


图 6-53　预测分类结果(1000 次)

对预测分类的输出结果(yn)如下：

```
yn =
1 ～ 15 列
       3     3     1     3     4     2     2     3     4     1     3     3     1     2     4
16 ～ 30 列
       2     4     3     4     2     2     3     3     1     1     4     1     3     3     3
```

(2) 将前 49 组数据(49×3)作为训练输入，后 10 组数据(10×3)作为测试输入，将得到的结果与实际分类进行比对。

设置迭代次数为 200 次，网络进化过程如图 6-54 所示，预测分类结果如图 6-55 所示。

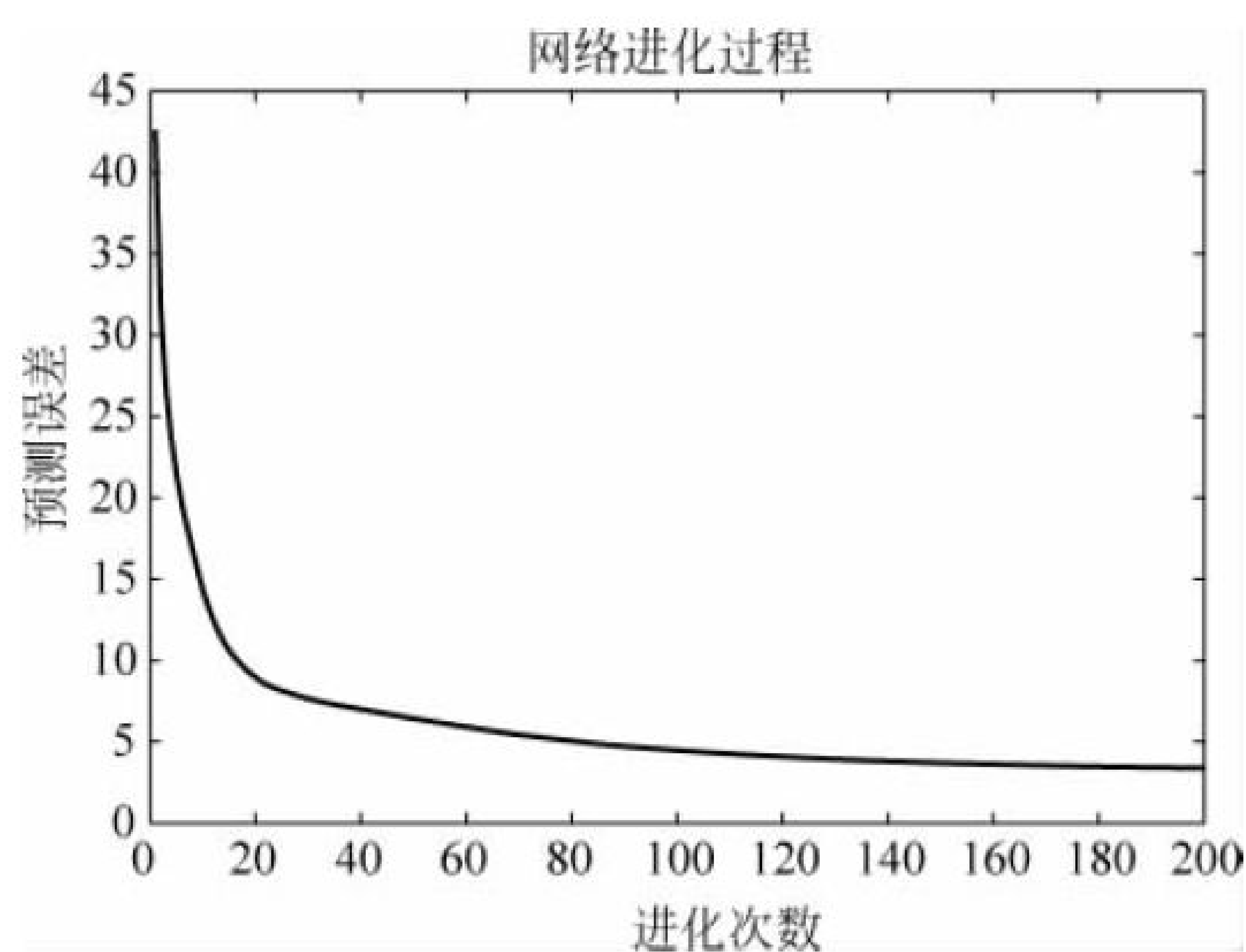


图 6-54 网络进化过程(200 次)

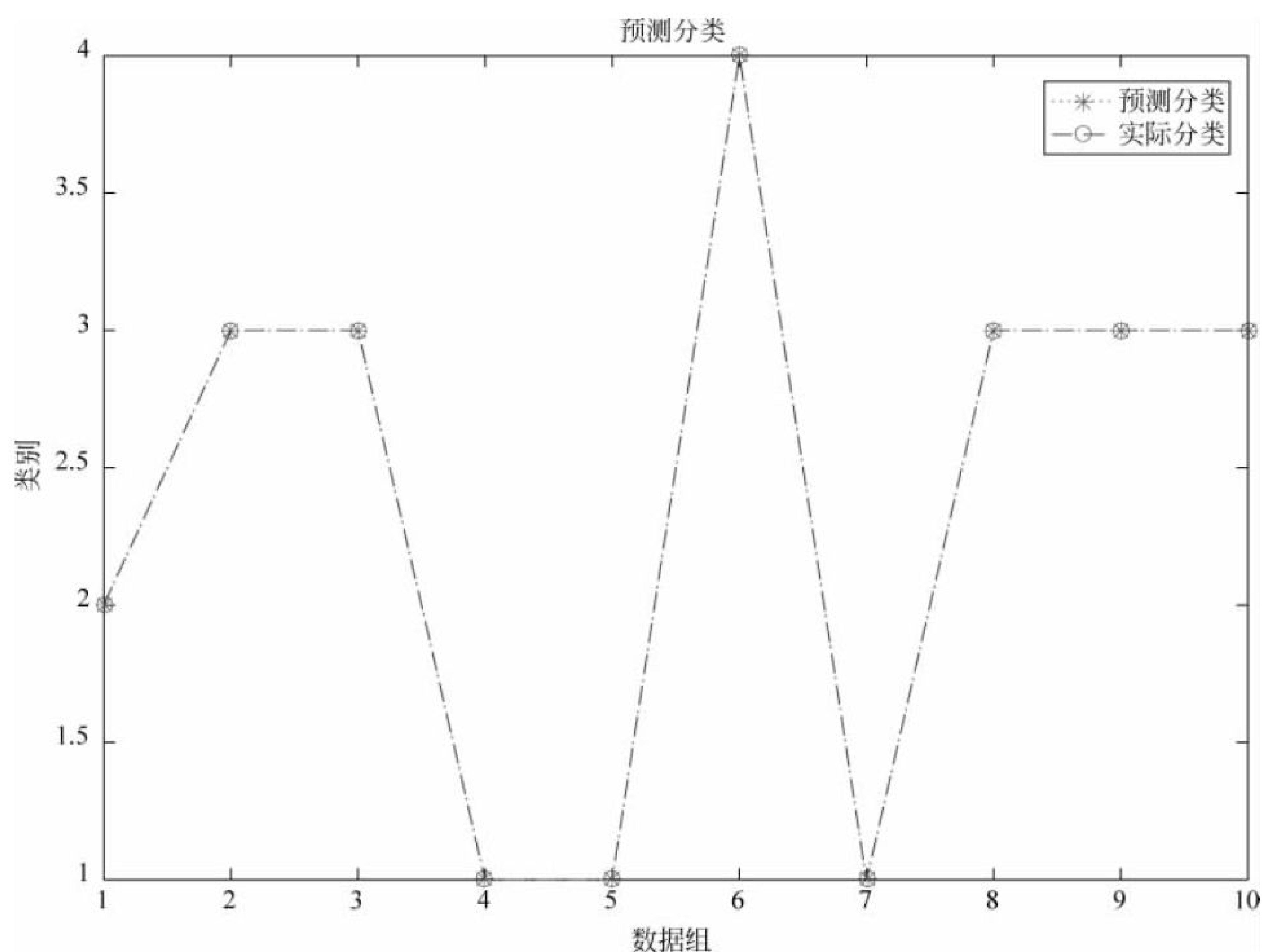


图 6-55 预测分类结果(200 次)

预测分类的输出结果(yn)如下：

```
yn =
       2     3     3     1     1     4     1     3     3     3
```

设置迭代次数为 300 次，网络进化过程如图 6-56 所示，预测分类结果如图 6-57 所示。

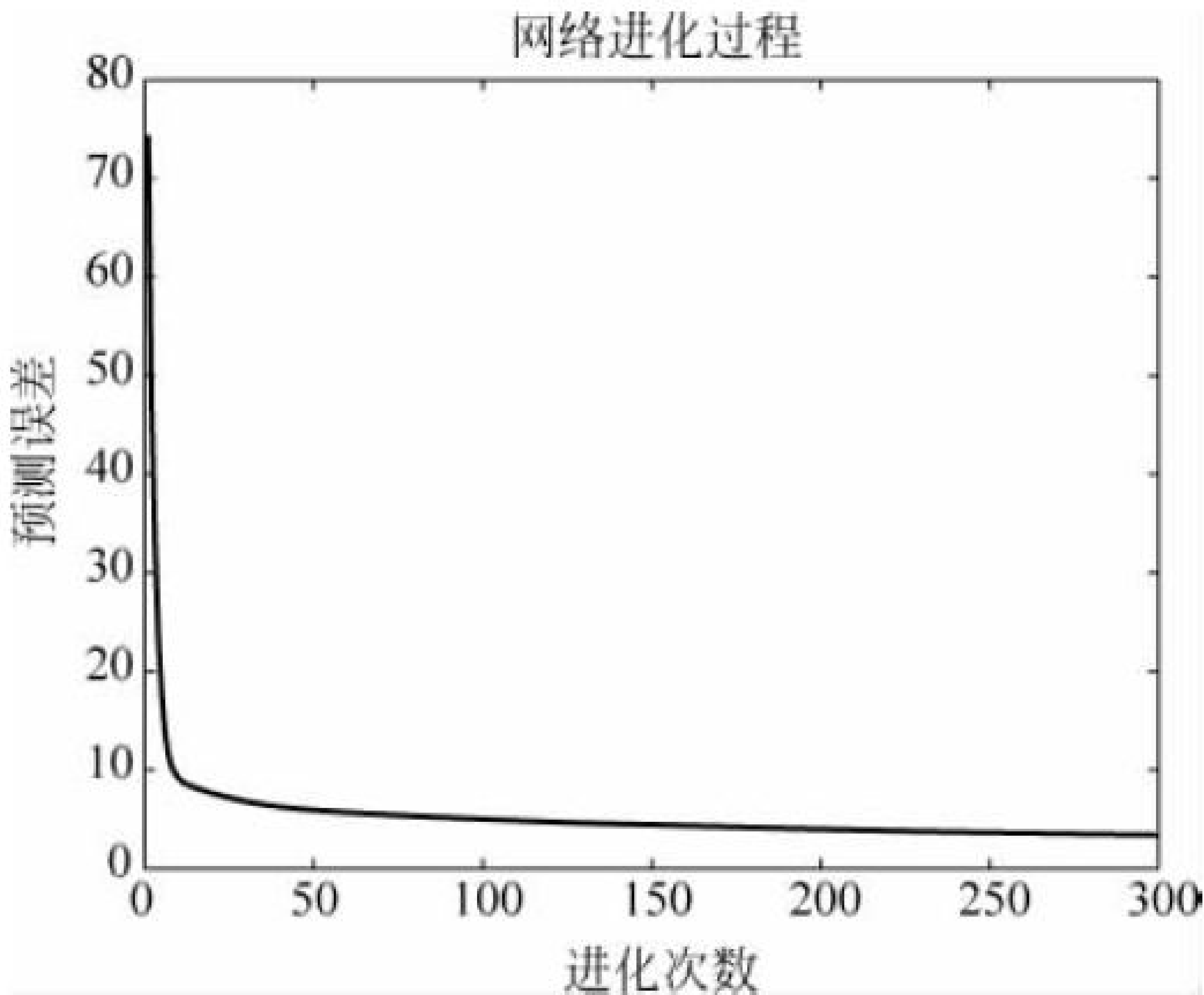


图 6-56　网络进化过程(300 次)

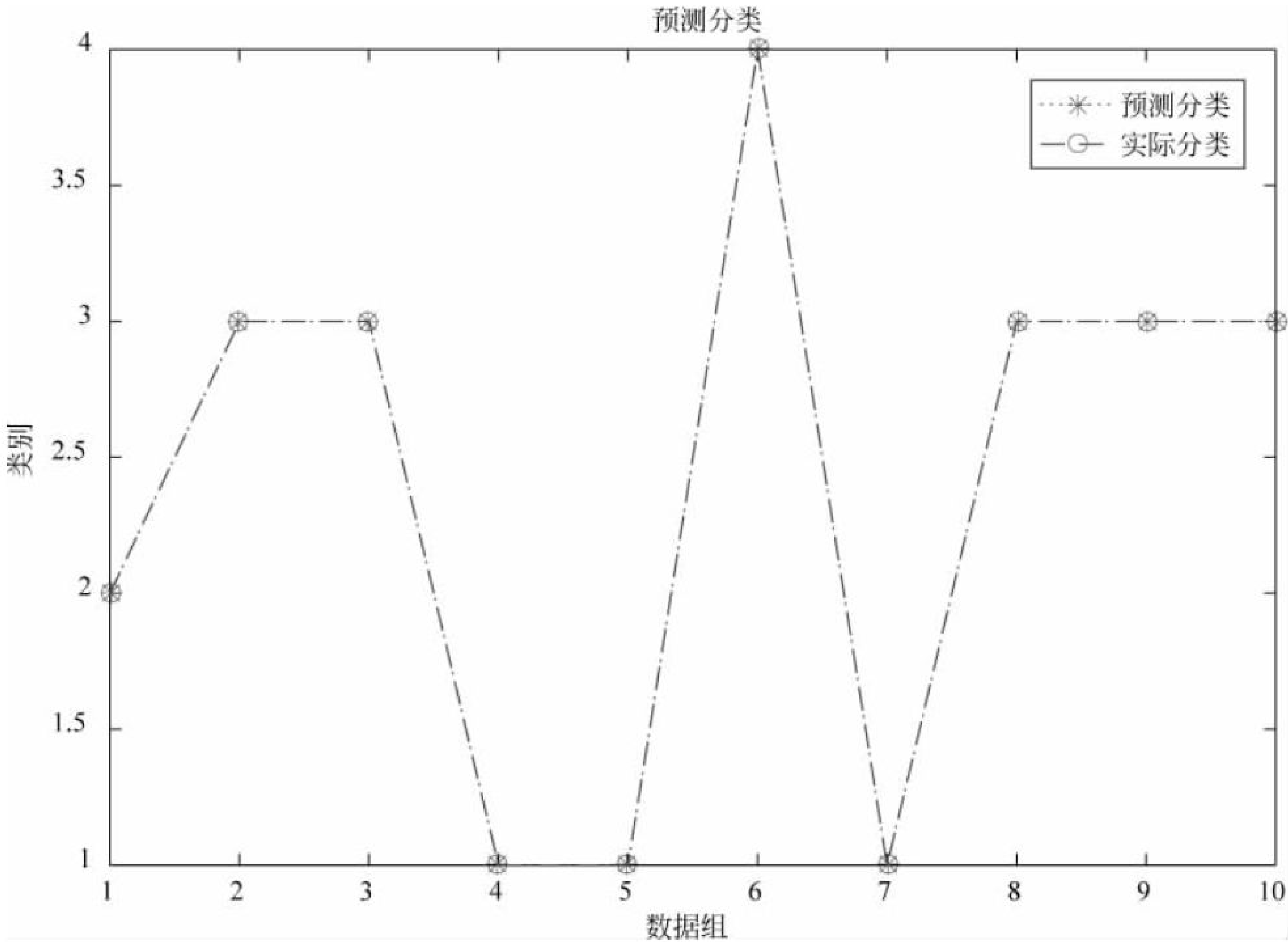


图 6-57　预测分类结果(300 次)

6.7.5 结论

通过酒瓶颜色分类实例可知,通过调整训练数据个数及迭代次数,小波神经网络可以得到理想的预测效果。小波神经网络具有良好的局部性特征,弥补了常规神经网络寻优速度慢、实时性差的缺点。实验结果表明,小波神经网络预测模型的预测精度明显优于 BP 神经网络,具有较高的应用价值。

6.8 其他形式的神经网络

在此只介绍竞争型人工神经网络、概率神经网络和 CPN 神经网络。

6.8.1 竞争型人工神经网络——自组织竞争

在实际的神经网络中,比如人的视网膜中,存在着一种“竞争”现象,即一个神经细胞兴奋后,通过它的分支会对周围其他神经细胞产生影响,使网络向更有利于竞争的方向调整,即一个“获胜”其他“全输”。

自组织竞争人工神经网络正是基于上述生物结构和现象形成的。神经网络分类器的学习方法,除了有导师或监督(supervised)、自监督(self-supervised)学习方法外,还有一种很重要的无导师或非监督(unsupervised)学习方法。自组织竞争系统就属于无导师型神经网络,这种自组织系统在待分类的模式无任何先验学习的情况下,很有用。它能够对输入模式进行自组织训练和判断,并将其最终分为不同的类型。

与 BP 网络相比,这种自组织自适应的学习能力进一步拓宽了人工神经网络在模式识别和分类方面的应用。另一方面,竞争学习网络的核心——竞争层,又是许多其他神经网络模型的重要组成部分。

1. 自组织竞争网络

竞争网络由单层神经元网络组成,其输入节点与输出节点之间为全互联结构。因为网络在学习中的竞争特性也表现在输出层上,所以竞争网络中的输出层又称为竞争层,而与输入节点相连的权值及其输入合称为输入层。其网络结构图如图 6-58 所示。

网络竞争层各神经元竞争对输入模式的响应机会,最后仅一个神经元成为竞争的胜利者,并对那些与获胜神经元有关的各连接权值朝着更有利于竞争的方向调整,获胜神经元表示输入模式的分类。

网络权值的调整公式为

$$\omega_{ij} = \omega_{ij} + a\left(\frac{x_i}{m} - \omega_{ij}\right) \tag{6-82}$$

该网络适用于模式识别和模式分类,尤其适用于具有大批相似数组的分类问题。

竞争网络适用于具有典型聚类特性的大量数据的辨识,但当遇到大量的具有概率分布

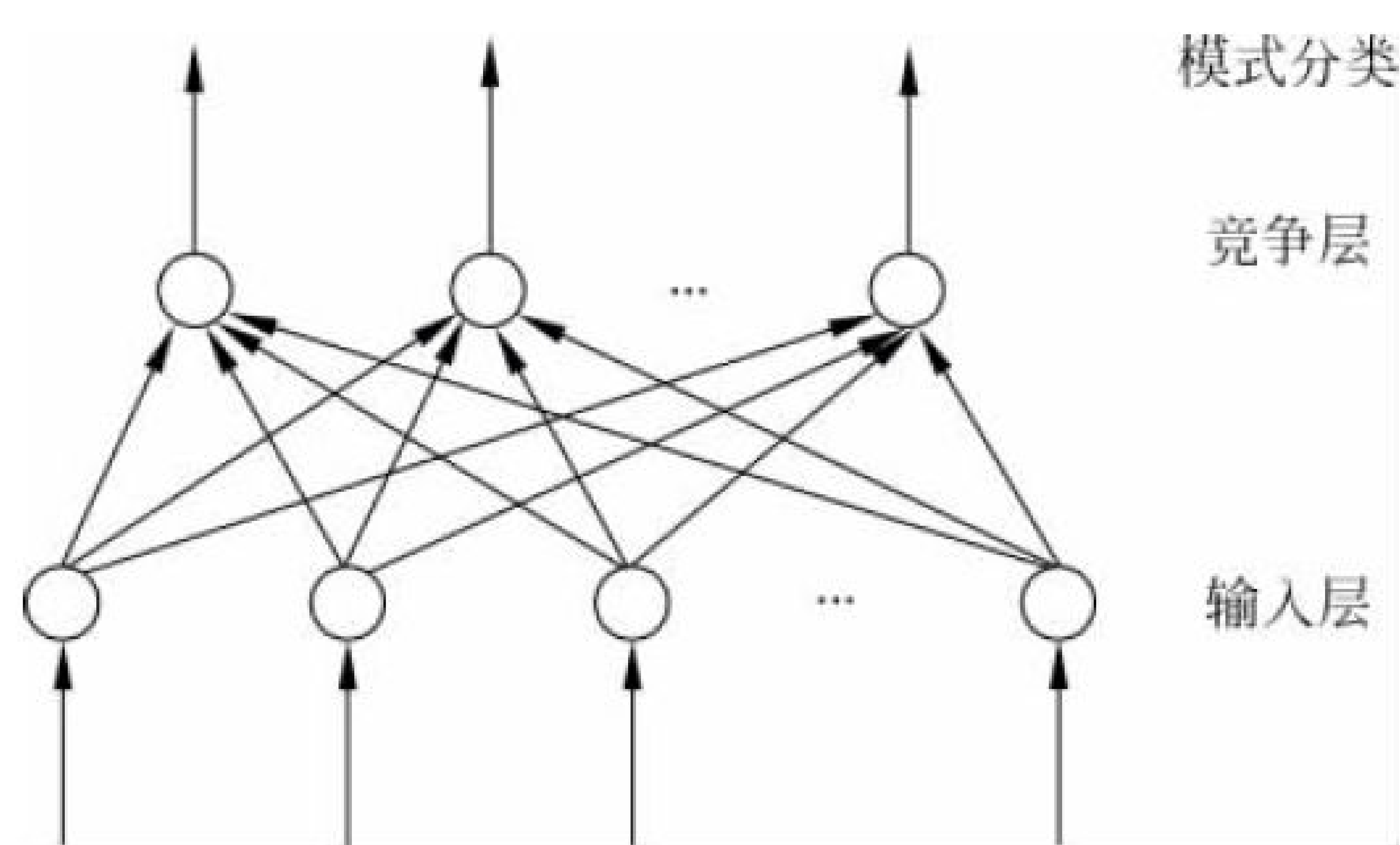


图 6-58　自组织竞争网络结构图

的输入向量时，竞争网络就无能为力了。

除了依靠竞争手段使神经元获胜的方法外，还有依靠抑制手段使神经元获胜的方法。当竞争层某个神经元的输入值大于其他所有神经元的输入值时，依靠其输出所具有的优势（即其输出值较其他的神经元大）通过抑制作用将其他神经元的输出值逐渐减小。这样，竞争层各神经元的输出就形成连续变化的模拟量。

设网络的输入向量为 $\boldsymbol{P}=[p_1 \quad p_2 \quad \cdots \quad p_r]$；对应网络的输出向量为 $\boldsymbol{A}=[a_1 \quad a_2 \quad \cdots \quad a_s]$。

由于竞争网络中含有两种权值，因此激活函数的加权输入和也分为两部分：来自输入节点的加权输入和 N 与来自竞争层内互相抑制的加权输入和 G。对于第 i 个神经元，有：
来自输入节点的加权输入和

$$n_i = \sum_{j=1}^{r} \omega_{ij} p_j \tag{6-83}$$

来自竞争层内互相抑制的加权输入和

$$g_i = \sum_{k \in D} \omega_{ij} f(a_k) \tag{6-84}$$

如果在竞争后，第 i 个节点“赢”了，则有 $a_k=1$，其中 $k=i$，而其他所有节点的输出均为 0，即 $a_k=0$，其中 $k=1,2,\cdots,s$ 且 $k \neq i$。此时，$g_i = \sum_{k=1}^{s} \omega_{ik} \cdot f(a_k) = \omega_{ii} > 0$。如果在竞争后，第 i 个节点“输”了，而“赢”的节点为 l，则

$$\begin{cases} a_k = 1, & k = l \\ a_k = 0, & k = 1,2,\cdots,s \text{ 且 } k \neq l \end{cases} \tag{6-85}$$

此时，$g_i = \sum_{k=1}^{s} \omega_{ik} \cdot f(a_k) = \omega_{il} < 0$。

所以，对整个网络的加权输入总和有下式成立

$$\begin{cases} s_l = n_l + \omega_{il}, & \text{对于“赢”的节点 } l \\ s_l = n_l - |\omega_{il}|, & \text{对于所有“赢”的节点 } i = 1,2,\cdots,s \text{ 且 } k \neq l \end{cases}$$

由此可以看出，经过竞争后只有获胜的那个节点的加权输入总和为最大。竞争网络的输出为

$$a_k = \begin{cases} 1, & s_k = \max(s_i) \quad i = 1,2,\cdots,s \\ 0, & \text{其他} \end{cases} \tag{6-86}$$

判断竞争网络节点胜负的结果时,可直接采用 n_i,即

$$n_{赢} = \max\left(\sum_{j=1}^{r}\omega_{ij}\boldsymbol{p}_j\right) \tag{6-87}$$

取偏差 b 为 0 是判定竞争网络获胜节点时的典型情况,偶尔也采用下式进行竞争结果的判定

$$n_{赢} = \max\left(\sum_{j=1}^{r}\omega_{ij}\boldsymbol{p}_j + b\right), \quad -1 < b < 0 \tag{6-88}$$

综上所述,竞争网络的激活函数使加权输入和为最大的节点赢得输出为 1,而其他神经元的输出皆为 0。

这个竞争过程可用 MATLAB 描述如下:

```
n = W * P;
[S,Q] = size(n);
x = n + b * ones(1,Q);
y = max(x);
for q = 1: Q
     % 找出最大加权输入和 y(q)所在的行;
    s = find(x(:,q) = y(q));
     % 令元素 a(z,q) = 1,其他值为 0;
    a(z(1),q) = 1;
end
```

这个竞争过程的程序已被包含在竞争激活函数 compet. m 之中:

```
A = compet(W * P,B);
```

网络创建函数 net=newc(PR,S,KLR,CLR),各参数含义如下:

PR:输入向量的范围,对未归一化的数据可以用语句 minmax(P)求出,$\boldsymbol{P}$ 为输入向量。

S:神经元个数,即分类目标数。

KLR:Kohonen 学习率,通常设为 0.1,默认为 0.01。

CLR:Conscience 学习率,默认为 0.001,通常为 0.001。

2. 自组织竞争网络应用于模式分类

以表 1-2 的三元色数据为例,按照颜色数据所表征的特点,将数据按各自所属的类别归类。其中,前 29 组数据已确定类别,后 30 组数据待确定类别。

使用自组织竞争网络对三元色数据进行分类,其 MATLAB 程序代码如下:

```
clear all
 % 训练样本
clc;
 % 训练样本
pConvert = importdata('SelfOrganizationCompetitiontrain.dat');
p = pConvert';
 % 创建网络
```

```
net = newc(minmax(P),4,0.1);
% 网络初始化
net = init(net);
% 设置训练次数,在消耗时间与精确度之间折中
net.trainParam.epochs = 200;
% 开始训练
net = train(net,P);
Y = sim(net,P)
% 分类数据转换输出
Yc = vec2ind(Y)
pause
% 待分类数据
dataConvert = importdata('SelfOrganizationCompetitionSimulation.dat');
data = dataConvert';
% 用训练好的自组织竞争网络分类样本数据
Y = sim(net,data);
Ys = vec2ind(Y)
```

由于竞争型网络采用的是无教师学习方式,没有期望输出,因此训练过程中不用设置判断网络是否结束的误差项。只要设置网络训练次数就可以了,并且在训练过程中也只显示训练次数。运行上述程序后,系统将显示运行过程,并给出聚类结果:

```
TRAINR,Epoch 0/200
TRAINR,Epoch 25/200
TRAINR,Epoch 50/200
TRAINR,Epoch 75/200
TRAINR,Epoch 100/200
TRAINR,Epoch 125/200
TRAINR,Epoch 150/200
TRAINR,Epoch 175/200
TRAINR,Epoch 200/200
TRAINR,Maximum epoch reached.

Yc  =
1 ～ 16 列
4    3    4    1    3    1    4    2    3    3    4    3    3    2
2    1
17 ～ 29 列
4    2    2    4    4    2    3    2    1    4    3    3    3
```

系统训练结束后,给出分类结果。由于竞争型网络采用的是无教师学习方式,因此其显示分类结果的方式与目标设置方式可能不同。这里采用统计法比较自组织竞争网络给出的结果与原始分类结果,如表 6-7 所示。

表 6-7 自组织竞争网络输出结果与原始分类结果对照表

原始分类结果统计	A	(数据序号) 4、6、16、25
	B	(数据序号) 8、14、15、18、19、22、24
	C	(数据序号) 1、3、7、11、17、20、21、26
	D	(数据序号) 2、5、9、10、12、13、23、27、28、29
自组织竞争网络分类结果统计	A	(数据序号) 4、6、16、25
	B	(数据序号) 8、14、15、18、19、22、24
	C	(数据序号) 1、3、7、11、17、20、21、26
	D	(数据序号) 2、5、9、10、12、13、23、27、28、29

从统计结果可知,自组织竞争网络输出结果与原始分类结果完全吻合。继续运行程序则可得到待分类样本数据的分类结果。

```
Ys =
1 ～ 15 列
4    2    4    3    2    3    4    1    2    2    4    2    2    1    1
16 ～ 30 列
3    4    1    1    4    4    1    2    1    3    4    2    2    2    4
31 ～ 45 列
4    3    4    2    1    1    4    2    3    4    4    3    1    2    1
46 ～ 49 列
2    4    2    1
```

6.8.2 竞争型人工神经网络——自组织特征映射神经网络

自组织特征映射神经网络(self-organizing feature mapping,SOM)也是无教师学习网络,主要用于对输入向量进行区域分类。其结构与基本竞争型神经网络很相似,与自组织竞争网络的不同之处是,SOM 网络不但识别属于区域邻近的区域,还研究输入向量的分布特性和拓扑结构。

自组织特征映射网络的基本思想是,最近的神经元相互激励,较远的相互抑制,更远的则又具有较弱的激励作用。SOM 网络的拓扑结构如图 6-59 所示。

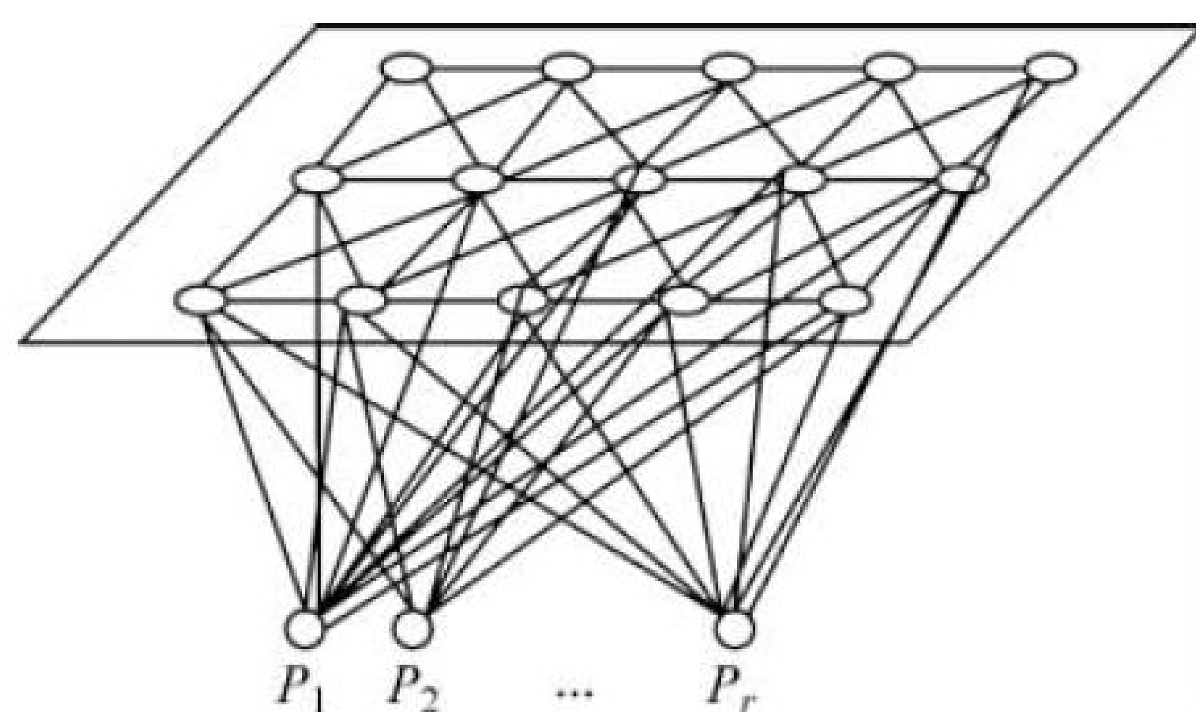


图 6-59 SOM 网络的拓扑结构

SOM 网络结构也是两层:输入层和竞争层。与基本竞争网络不同之处是其竞争层可以由一维或二维网络矩阵方式组成,并且权值修正的策略也不同。一维网络结构与基本竞

争学习网络相同。

SOM 网络可以用来识别获胜神经元 i。不同的是，自组织竞争网络只修正获胜神经元，而 SOM 网络依据 Kohonen 规则，同时修正获胜神经元附近区域 $Ni(d)$ 内的所有神经元。SOM 网络神经元邻域示意图如图 6-60 所示。

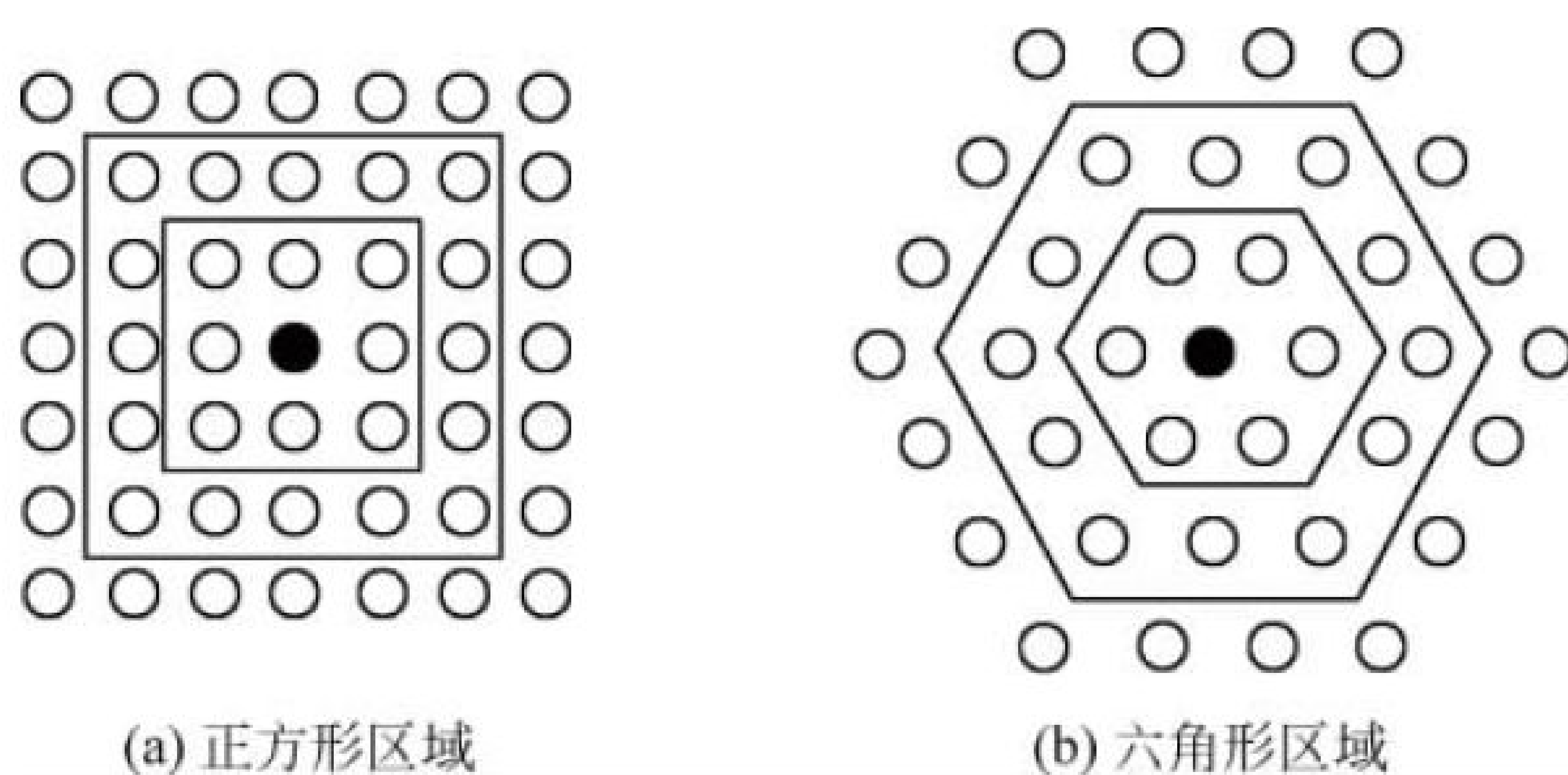


图 6-60 SOM 网络神经元邻域示意图

在 MATLAB 工具箱中有一个求获胜神经元的邻域的函数，在二维竞争层中，邻域函数为 neighb2d.m，用法如下：

```
Np = [x y];
in = neighb2d(I,Np,N);
```

网络创建函数如下：

```
net = newsom(PR,[D1,D2 … ],TFCN,DFCN)
```

各参数含义如下。

PR：输入向量的范围。

Di：第 i 维神经元个数，与分类目标数有关。

TFCN：布局函数，默认为 hextop(六角形布局)。

DFCN：距离函数，默认为 linkdist(连接距离)。

其他：排序调整等学习率，通常使用默认值。

使用自组织特征映射神经网络将三元色数据按照颜色数据所表征的特点归类。其 MATLAB 程序代码如下：

```
clear;
clc;
%训练样本
pConvert = importdata('SelfOrganizationCompetitiontrain.dat');
p = pConvert';
net = newsom(minmax(p),[4 1]);
%神经元排列为[1 4]时结果相同,只是神经元的位置改变了
%设置网络训练次数
net.trainParam.epochs = 200;
```

```
%开始训练
net = train(net,p);
%绘制网络的神经元分布图
plotsom(net.layers{1}.positions);
%用训练好的自组织竞争网络对样本点分类
Y = sim(net,p);
%分类数据转换输出
Yt = vec2ind(Y)
pause
%待分类数据
dataConvert = importdata('SelfOrganizationCompetitionSimulation.dat');
data = dataConvert';
%用训练好的自组织竞争网络分类样本数据
Y = sim(net,data);
Ys = vec2ind(Y)
```

由于自组织特征映射神经网络采用的是无教师学习方式,没有期望输出,因此训练过程中不用设置判断网络是否结束的误差项。只要设置网络训练次数就可以了,并且在训练过程中也只显示训练次数。运行上述程序后,系统显示运行过程,并给出聚类结果:

```
TRAINR,Epoch 0/200
TRAINR,Epoch 25/200
TRAINR,Epoch 50/200
TRAINR,Epoch 75/200
TRAINR,Epoch 100/200
TRAINR,Epoch 125/200
TRAINR,Epoch 150/200
TRAINR,Epoch 175/200
TRAINR,Epoch 200/200
TRAINR,Maximum epoch reached.
Yt =
1 ～ 16 列
2    4    2    3    4    3    2    1    4    4    2    4    4    1
1    3
17 ～ 29 列
2    1    1    2    2    1    4    1    3    2    4    4    4
```

系统训练结束后,给出分类结果。由于竞争型网络采用的是无教师学习方式,因此其显示分类结果的方式与目标设置方式可能不同。这里采用统计法比较自组织竞争网络给出的结果与原始分类结果,如表6-8所示。

表6-8 自组织特征映射神经网络输出结果与原始分类结果对照

原始分类结果统计	A	(数据序号)4、6、16、25
	B	(数据序号)8、14、15、18、19、22、24
	C	(数据序号)1、3、7、11、17、20、21、26
	D	(数据序号)2、5、9、10、12、13、23、27、28、29
自组织特征映射神经网络分类结果统计	A	(数据序号)4、6、16、25
	B	(数据序号)8、14、15、18、19、22、24
	C	(数据序号)1、3、7、11、17、20、21、26
	D	(数据序号)2、5、9、10、12、13、23、27、28、29

从统计结果可知,自组织特征映射神经网络输出结果与原始分类结果完全吻合。网络的神经元分布图如图 6-61 所示。

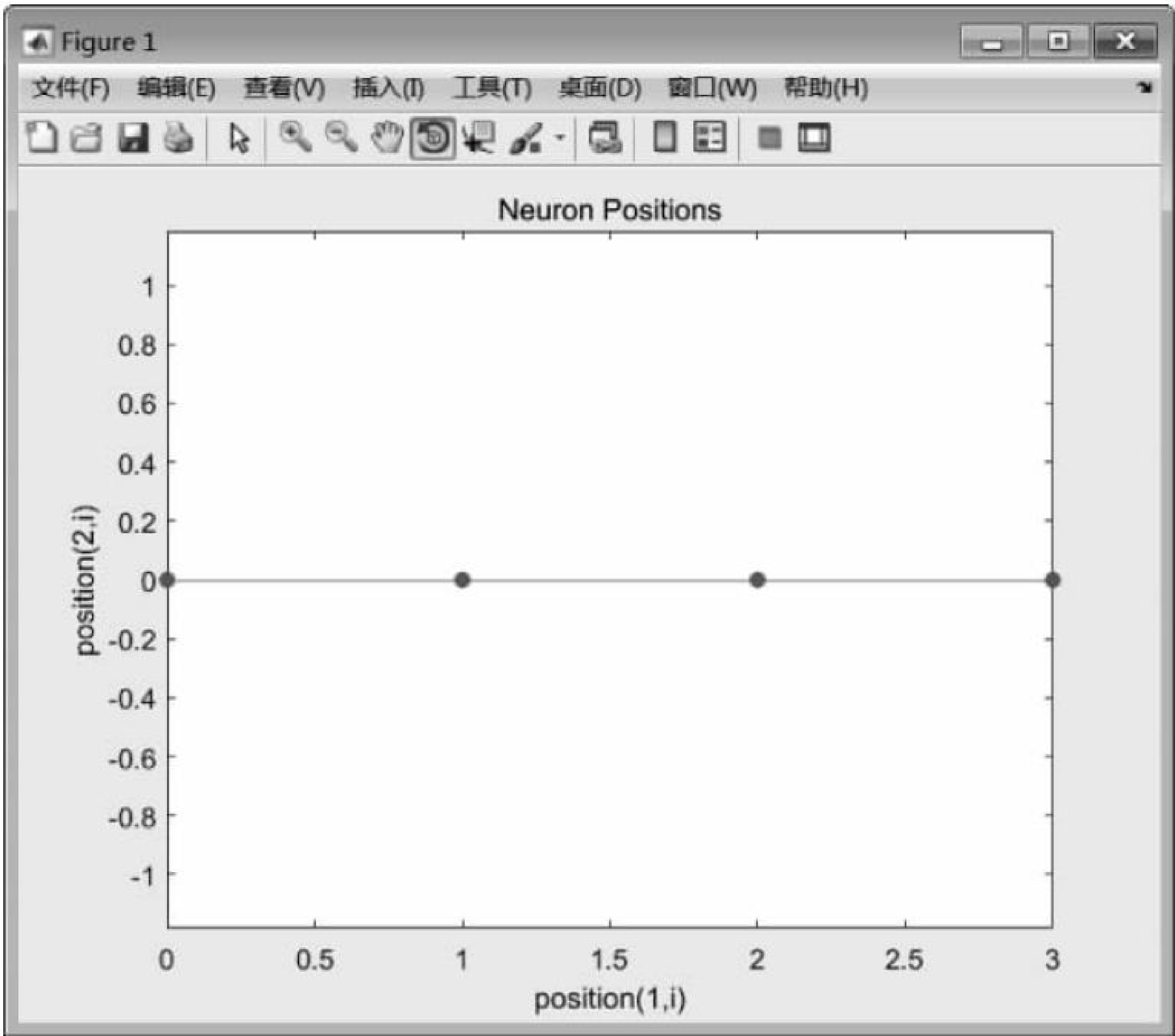


图 6-61 网络的神经元分布图

继续运行程序则可得到待分类样本数据的分类结果(调整显示分类结果方式后):

```
Ys =
1 ～ 15 列
2    4    2    3    4    3    2    1    4    4    2    4    4    1    1
16 ～ 30 列
3    2    1    1    2    2    1    4    1    3    2    4    4    4    2
31 ～ 45 列
2    3    2    4    1    1    2    4    3    2    2    3    1    4    1
46 ～ 49 列
4    2    4    1
```

6.8.3 竞争型人工神经网络——学习向量量化神经网络

学习向量量化(learning vector quantization,LVQ)神经网络是一种有教师训练竞争层的方法,主要用来进行向量识别。LVQ 神经网络是两层的网络结构:第一层为竞争层,和前面的自组织竞争网络的竞争层功能相似,用于对输入向量分类;第二层为线性层,将竞争层传递过来的分类信息转换为使用者所定义的期望类别。

通常将竞争层学习得到的类称为子类,经线性层的类称为期望类别(目标类)。

1. 学习向量量化的基本模型

学习向量量化网络和自组织映射网络具有非常类似的网络结构，如图 6-62 所示。网络由输入层和输出层组成，输入层具有 N 个输入节点，接受输入向量 $\boldsymbol{X}=[x_1,x_2,\cdots,x_N]^{\mathrm{T}}$。输出层有 M 个神经元，呈一维线性排列。学习向量量化没有在输出层引入拓扑结构，因此在网络学习中也不再有获胜邻域的概念。输入节点和输出层神经元通过权值向量 $\boldsymbol{W}=[\omega_{11},\cdots,\omega_{ij},\cdots,\omega_{MN}]^{\mathrm{T}}(i=1,2,\cdots,M;\ j=1,2,\cdots,N)$实现完全互连。其中任一神经元用 i 表示，其输入为输入向量和权值向量的内积 $u_i=\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}=\sum_{j=1}^{N}\omega_{ij}x_j(i=1,2,\cdots,M)$。神经元的输出为 $v_i=f(u_i)$，其中 $f(\cdot)$为神经元激励函数，一般取为线性函数。

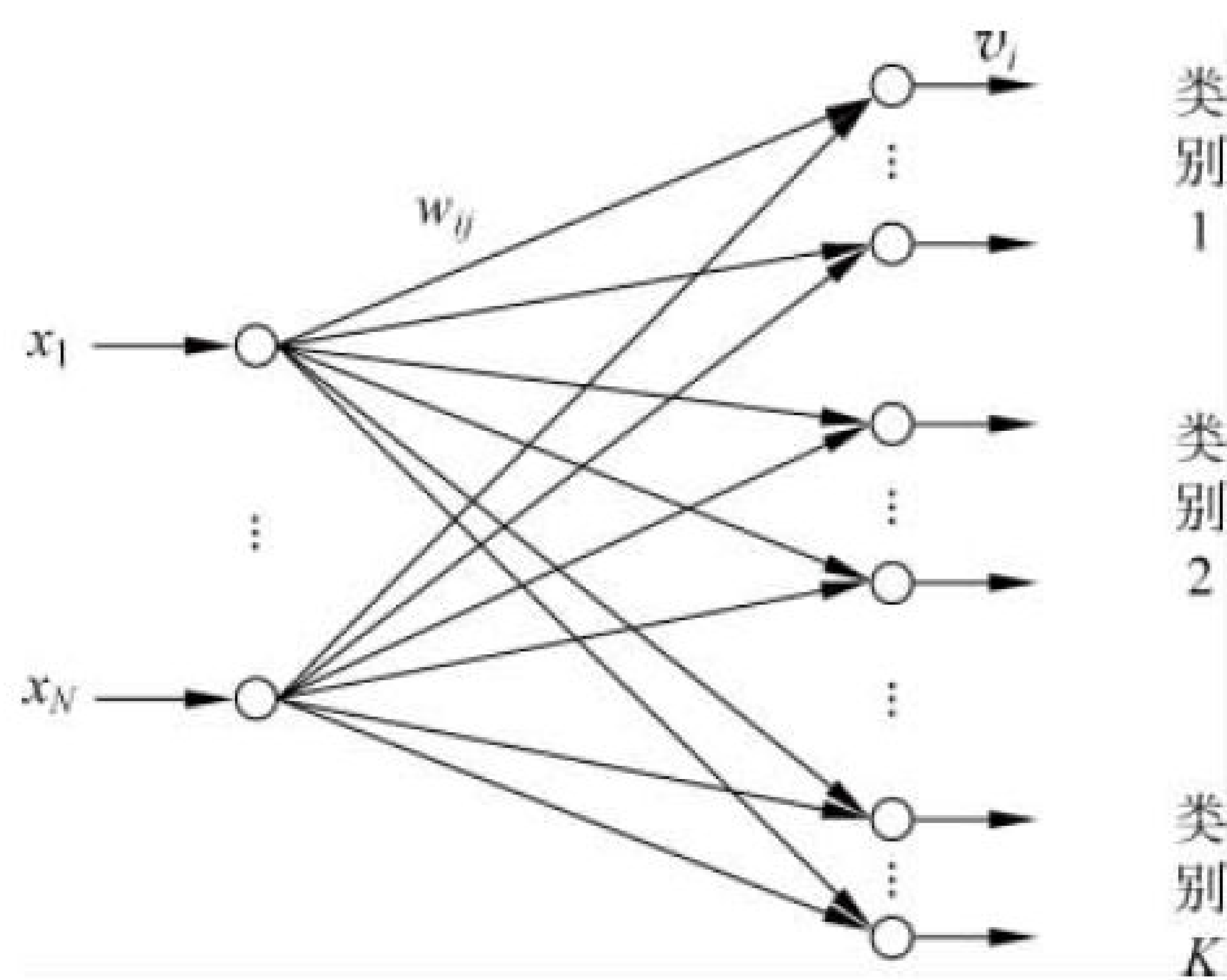


图 6-62　学习向量量化网络模型

需要强调的是，在学习向量量化中输出神经元被预先指定了类别。输出神经元被分为 K 组而代表 K 个类别，每个神经元所代表的类别在网络训练前被指定好。

2. 网络的学习算法

(1) 设置变量和参量

$\boldsymbol{X}(n)=[x_1(n),x_2(n),\cdots,x_N(n)]^{\mathrm{T}}$，为输入向量，或称训练样本。

$\boldsymbol{W}_i(n)=[\omega_{11}(n),\cdots,\omega_{ij}(n),\cdots,\omega_{MN}(n)]^{\mathrm{T}}$，为权值向量，$i=1,2\cdots,M,j=1,2,\cdots,N$。

选择学习速度的函数 $\eta(n)$，n 为迭代次数，K 为迭代总次数。

(2) 初始化权值向量 $\boldsymbol{W}_i(0)$及学习速率 $\eta(0)$。

(3) 从训练集合中选取输入向量 $\boldsymbol{X}$。

(4) 寻找获胜神经元 c

$$\|\boldsymbol{X}-\boldsymbol{W}_c\|=\min\|\boldsymbol{X}-\boldsymbol{W}_i\|,\quad i=1,2,\cdots,M \tag{6-89}$$

(5) 判断分类是否正确。根据以下规则调整获胜神经元的权值向量：

用 L_{wc}代表与获胜神经元权值向量相联系的类，用 L_{x^i}代表与输入向量相联系的类。

如果 $L_{x^i}=L_{\mathrm{wc}}$，则 $\boldsymbol{W}_c(n+1)=\boldsymbol{W}_c(n)+\eta(n)[\boldsymbol{X}-\boldsymbol{W}_c(n)]$；否则，若 $L_{x^i}\neq L_{\mathrm{wc}}$，则

$$\boldsymbol{W}_c(n+1)=\boldsymbol{W}_c(n)-\eta(n)[\boldsymbol{X}-\boldsymbol{W}_c(n)] \tag{6-90}$$

对于其他神经元，保持权值向量不变。

（6）调整学习速率 $\eta(n)$

$$\eta = \eta(0)\left(1 - \frac{n}{N}\right) \tag{6-91}$$

（7）判断迭代次数 n 是否超过 K：如果 $n \leqslant K$，就将 n 值增加 1，转到（3）；否则结束迭代过程。

注意：在算法中，必须保证学习常数 $\eta(n)$ 随着迭代次数 n 的增加单调减小。例如，$\eta(n)$ 被初始化为 0.1 或更小，之后随着 n 的增加而减小。在算法中我们没有对权值向量和输入向量进行归一化处理，这是因为网络直接把权值向量和输入向量的欧氏距离最小作为判断竞争获胜的条件。

3. LVQ 的实现代码

网络创建函数如下：

```
net = newlvq(PR,S1,PC,LR,LF)
```

函数各参数含义如下。

PR：输入向量的范围。

S：竞争层神经元的个数，可设置为分类目标数。

PC：线性层输出类别比率向量。

LR：学习率，默认为 0.01。

LF：学习函数，默认为 learnlv1。

使用学习向量量化神经网络将三元色数据按照颜色数据所表征的特点归类。其 MATLAB 实现程序代码如下：

```
clear;
clc;
%训练样本
pConvert = importdata ('C:\Users\Administrator\Desktop\ln\SelfOrganizationtrain.dat');
p = pConvert';
%训练样本的目标矩阵
t = importdata ('C:\Users\Administrator\Desktop\ln\SelfOrganizationtarget.dat');
t = t';
%向量转换
t = ind2vec(t);
%创建网络
net = newlvq(minmax(p),4,[.32 .29 .25 .14]);
%开始训练
net = train(net,p,t);
%用训练好的自组织竞争网络对样本点分类
Y = sim(net,p);
%分类数据转换输出
Yt = vec2ind(Y)
pause
%待分类数据
```

```
dataConvert = importdata('C:\Users\Administrator\Desktop\ln\SelfOrganizationSimulation.dat');
data = dataConvert';
% 用训练好的自组织竞争网络分类样本数据
Y = sim(net,data);
Ys = vec2ind(Y)
```

运行上述程序后,系统显示运行过程,并给出聚类结果:

```
TRAINR,Epoch 0/100
TRAINR,Epoch 4/100
TRAINR,Performance goal met.
Yt =
1 ～ 15 列
3    4    3    1    4    1    3    2    4    4    3    4    4    2    2
16 ～ 29 列
1    3    2    2    3    3    2    4    2    1    3    4    4    4
```

图 6-63 为神经网络训练模块,在这里可以查看训练结果、训练状态等。训练后即可达到对误差的要求,结果如图 6-64 所示。

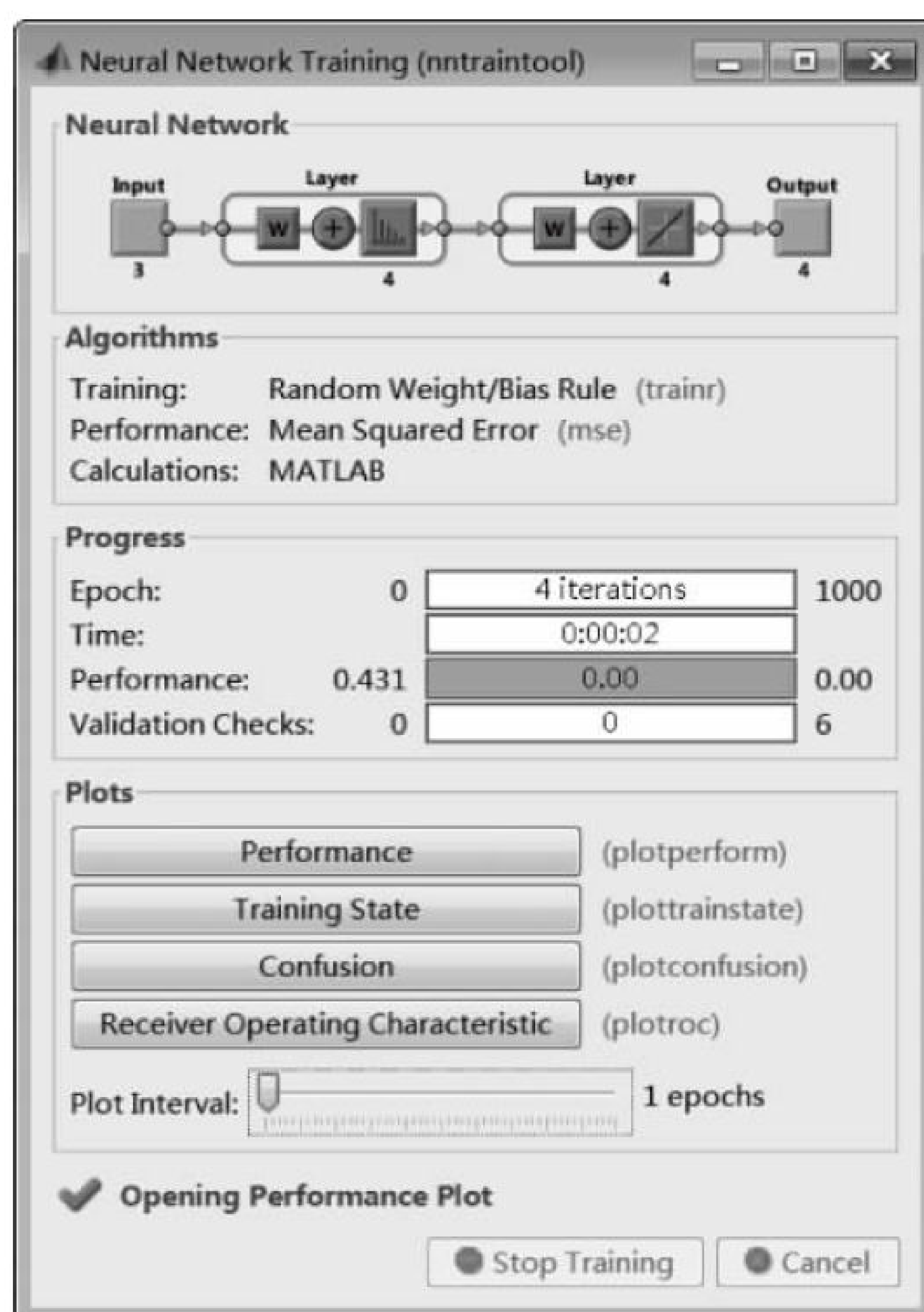


图 6-63　神经网络训练图

训练后的 LVQ 网络对训练数据进行分类后的结果与目标结果对比如表 6-9 所示。

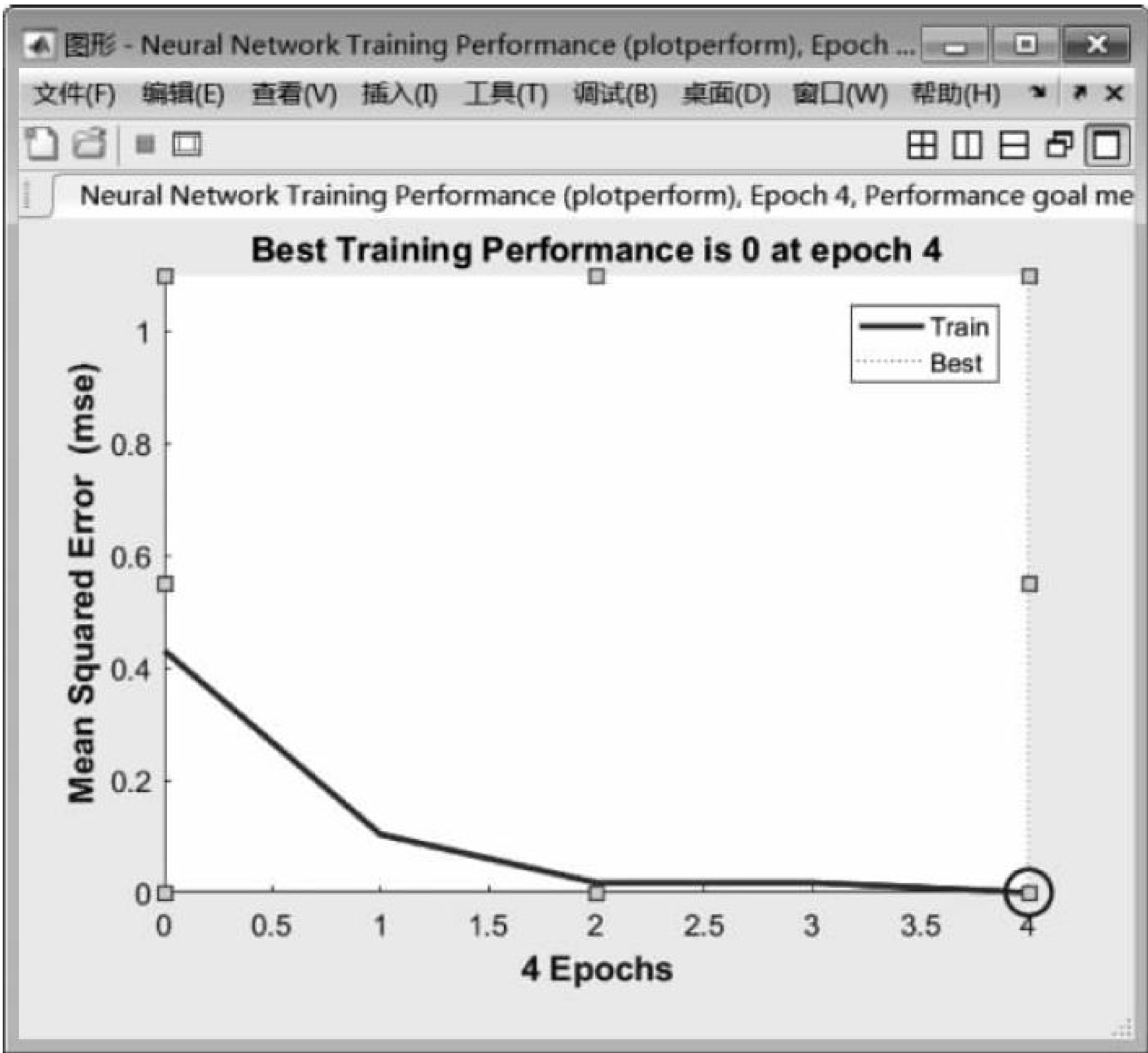


图 6-64 学习向量量化神经网络训练结果图

表 6-9 训练后的 LVQ 网络对训练数据进行分类后的结果与目标结果对比

序 号	A	B	C	原始分类结果	LVQ 网络分类结果
1	1739.94	1675.15	2395.96	3	3
2	373.3	3087.05	2429.47	4	4
3	1756.77	1652	1514.98	3	3
4	864.45	1647.31	2665.9	1	1
5	222.85	3059.54	2002.33	4	4
6	877.88	2031.66	3071.18	1	1
7	1803.58	1583.12	2163.05	3	3
8	2352.12	2557.04	1411.53	2	2
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
11	1571.17	1731.04	1735.33	3	3
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
16	1418.79	1775.89	2772.9	1	1
17	1845.59	1918.81	2226.49	3	3
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
20	1692.62	1867.5	2108.97	3	3
21	1680.67	1575.78	1725.1	3	3

续表

序　号	A	B	C	原始分类结果	LVQ 网络分类结果
22	2802.88	3017.11	1984.98	2	2
23	172.78	3084.49	2328.65	4	4
24	2063.54	3199.76	1257.21	2	2
25	1449.58	1641.58	3405.12	1	1
26	1651.52	1713.28	1570.38	3	3
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4

由表 6-9 可见，训练后的 LVQ 网络对训练数据进行分类后的结果与目标结果完全吻合，表明 LVQ 网络训练效果良好。继续运行程序则可得到待分类样本数据的分类结果：

```
Ys =
1～15 列
3    4    3    1    4    1    3    2    4    4    3    4    4    2    2
16～30 列
1    3    2    2    3    3    2    4    2    1    3    4    4    4    3
31～45 列
3    4    3    4    2    2    3    4    1    3    3    1    2    4    2
46～49 列
4    3    4    2
```

比较三种竞争型人工神经网络分类器的分类结果：

```
Ys =(自组织竞争调整显示方式后的输出结果)
3    3    1    3    4    2    2    3    4    1    3    3    1
2    4    2    4    3    4    2    2    3    3    1    1    4
1    3    3
Ys =(SOM 调整显示方式后的输出结果)
3    3    1    3    4    2    2    3    4    1    3    3    1
2    4    2    4    3    4    2    2    3    3    1    1    4
1    3    3    3
Ys =(LVQ)
3    3    4    3    4    2    2    3    4    1    3    3    1
2    4    2    4    3    4    2    2    3    3    1    1    4
1    3    3    3
```

经对比可知，基本竞争型网络与 SOM 网络的分类结果相同，而与 LVQ 网络第 3 组数据的分类结果不同，与人工分类对比，发现 LVQ 网络出错。前两种网络对数据的分类完全正确。

调整 LVQ 网络后用训练样本进行训练，但分类结果没有改变，与原分类结果相同（因为该网络对其他数据的分类结果正确，所以未对网络参数做调整），原因为 LVQ 网络的竞

争层识别的类别仅与输入向量间的距离有关。如果两个输入向量类似，竞争层就可能将其归为一类，竞争层的设计并没有严格界定不能将任意两个输入向量归于同一类。

6.8.4 概率神经网络

1. 概率神经网络结构和工作原理

径向基神经元还可以和竞争神经元一起共同组建概率神经网络（probabilistic neural network，PNN）。概率神经网络经常用于解决分类问题。PNN 的结构图如图 6-65 所示。

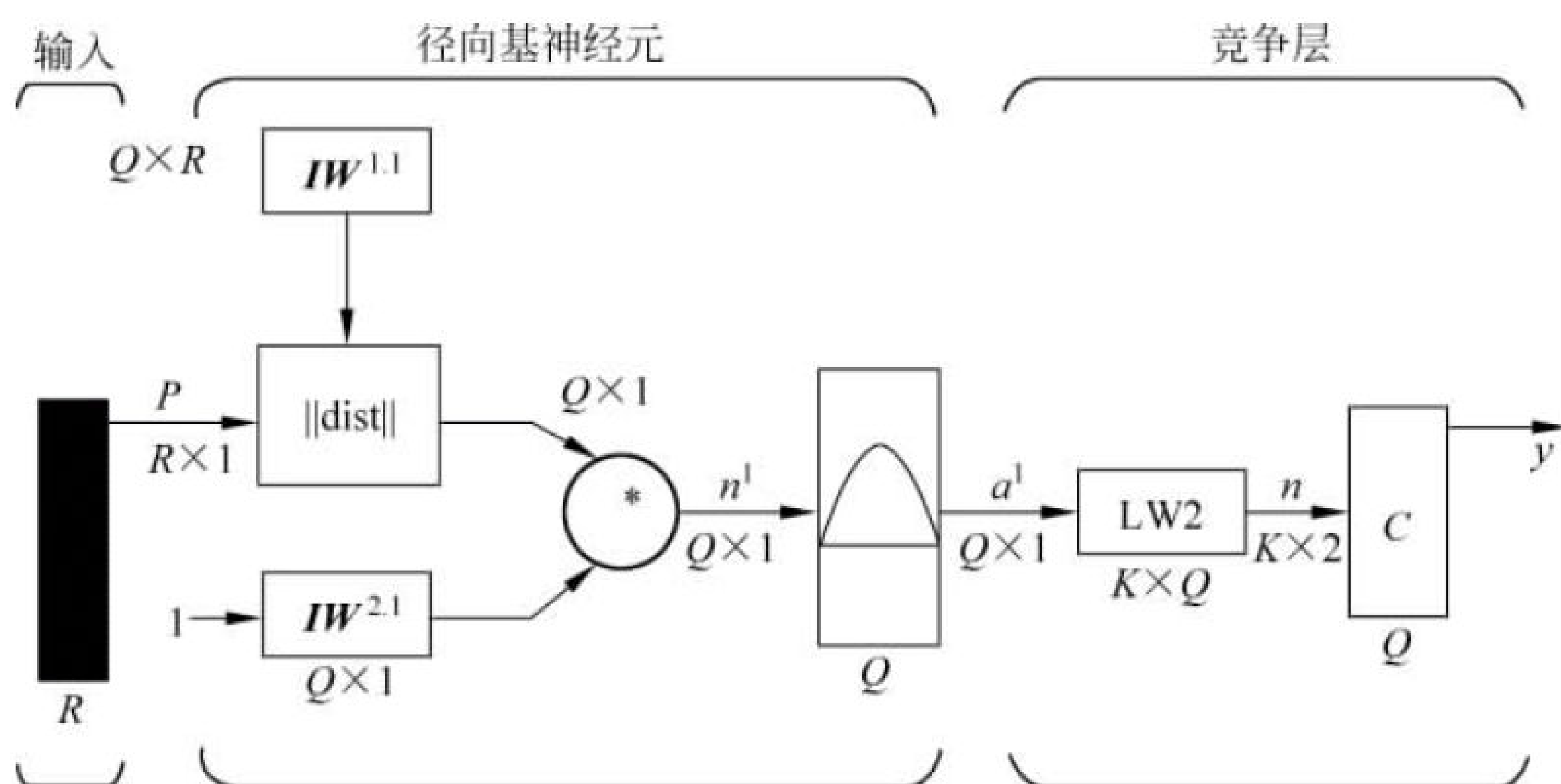


图 6-65 PNN 的结构图

PNN 的结构图和径向基函数网络结构类似，只是在第二层有些差异。其中，

$$a_i^1 = \mathrm{radbas}(\| {}_i\boldsymbol{IW}^{1,1} - p \| b_i^1) \tag{6-92}$$

$$y = a^2 = \mathrm{compet}(\boldsymbol{IW}^{2,1} - a^2) \tag{6-93}$$

$\boldsymbol{a}_i^1$ 为向量 $\boldsymbol{a}^1$ 的第 i 个元素；${}_i\boldsymbol{IW}^{1,1}$ 为权值矩阵 $\boldsymbol{IW}^{1,1}$ 的第 i 个行向量；R＝输入向量元素的数目＝第一层神经元的数目；Q＝输入向量类别的数目＝第二层神经元的数目。

PNN 的第一层仍与径向基函数网络的第一层类似。首先计算输入向量与训练样本之间的距离，第一层输出向量表示输入向量与训练样本之间的接近程度。将第二层与输入向量相关的所有类别综合在一起，网络输出为表示概率的向量，最后通过第二层的竞争（Compete）传递函数进行取舍，概率最大值的那一类为 1，其他类别用 0 表示。

假设输入期望值样本数量的数目为 Q，期望为 K 维向量，表示类别只有一个元素为 1，其余均为 0。

PNN 第一层的输入权值 $\boldsymbol{IW}^{1,1}$ 为输入样本的转置矩阵 $\boldsymbol{P}^{\mathrm{T}}$，经过 ‖dist‖ 计算，第一层输出向量表示输入向量与样本向量接近的程度，然后与阈值向量相乘，再经过径向传递函数计算。输入向量与哪个样本最接近，则 $\boldsymbol{a}^1$ 相对应的几个元素均为 1。

第二层权值 $\boldsymbol{IW}^{2,1}$ 设定为期望值向量矩阵 $\boldsymbol{T}$，每个行向量只有一个元素为 1，代表相应的类别，其余元素均为 0，然后计算乘积 $\boldsymbol{T}a^1$。最后通过第二层传递函数竞争计算得到 n^2，较大的元素取为 1，其余为 0。至此 PNN 即可完成对输入向量的分类。

2. 概率神经网络分类器的实现步骤

(1) 提取样本数据,样本数据如表 1-2 所示。其中,前 29 组数据已确定类别,后 30 组数据待确定类别。

取前 29 组数据作为训练样本。为了编程方便,先对这 29 组数据按类别进行升序排序。重新排序后的数据如表 1-1 所示。

(2) 调用 ind2vec 函数,将类别向量转换为 PNN 可以使用的目标向量。

(3) 设定径向基函数的散布常数。

在概率神经网络分类器的设计中,同样需要设定径向基函数的散布常数。同样,散布常数是分类器的设计中非常重要的参数,默认时为 1。这里散布常数确定在 30～260。

(4) 调用函数 newpnn,构建并训练 PNN。

在 MATLAB 中,构建概率神经网络的函数文件为 newpnn(),newpnn 文件的定义如下:

```
net = newpnn(P,T,SPREAD)
```

各个参数的定义如下。

P:Q 个输入向量的 $R\times Q$ 维矩阵。这里 $Q=29$,$R=3$。

T:Q 个目标类别向量的 $S\times Q$ 维矩阵。这里 $S=1$。

SPREAD:径向基函数的散布常数,默认时为 1.0。

(5) 调用 sim,测试 PNN 的训练效果。

(6) 再次调用 sim,识别测试样本所属类别。

(7) 调用 vec2ind 函数将分类结果转换为容易识别的类别向量。

3. 实现概率神经网络分类器

使用 MATLAB 软件实现概率神经网络分类器,程序代码如下:

```
clear;
clc;
%网络训练样本
pConvert = importdata('C:\Users\Administrator\Desktop\RBF\rbf_train_sample_data.dat');
p = pConvert';
%训练样本的目标矩阵
t = importdata('C:\Users\Administrator\Desktop\RBF\rbf_train_target_data.dat');
plot3(p(1,:),p(2,:),p(3,:),'o');
grid;box;
for i = 1:29,text(p(1,i),p(2,i),p(3,i),sprintf(' %g',t(i))),end
%以图形方式输出训练样本点
hold off
f = t';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
```

```
index4 = find(f == 4);
line(p(1,index1),p(2,index1),p(3,index1),'linestyle','none','marker',' * ','color','g');
line(p(1,index2),p(2,index2),p(3,index2),'linestyle','none','marker',' * ','color','r');
line(p(1,index3),p(2,index3),p(3,index3),'linestyle','none','marker',' + ','color','b');
line(p(1,index4),p(2,index4),p(3,index4),'linestyle','none','marker',' + ','color','y');
box;grid on;hold on;
axis([0 3500 0 3500 0 3500]);
title('训练用样本及其类别');
xlabel('A');
ylabel('B');
zlabel('C');
pause
t = ind2vec(t);
spread = 30;
% PNN 网络的创建和训练过程
net = newpnn(p,t,spread);
A = sim(net,p);
Ac = vec2ind(A)
plot3(p(1,:),p(2,:),p(3,:),'.'),grid;box;
axis([0 3500 0 3500 0 3500])
for i = 1:29,text(p(1,i),p(2,i),p(3,i),sprintf(' %g',Ac(i))),end
% 以图形方式输出训练结果
hold off
f = Ac';
index1 = find(f == 1);
index2 = find(f == 2);
index3 = find(f == 3);
index4 = find(f == 4);
line(p(1,index1),p(2,index1),p(3,index1),'linestyle','none','marker',' * ','color','g');
line(p(1,index2),p(2,index2),p(3,index2),'linestyle','none','marker',' * ','color','r');
line(p(1,index3),p(2,index3),p(3,index3),'linestyle','none','marker',' + ','color','b');
line(p(1,index4),p(2,index4),p(3,index4),'linestyle','none','marker',' + ','color','y');
box;grid on;hold on;
title('网络训练结果');
xlabel('A');
ylabel('B');
zlabel('C');
pause
% 对待分类样本进行分类
pConvert = importdata('C:\Users\Administrator\Desktop\RBF\rbf_simulate_data.dat');
p = pConvert';
a = sim(net,p);
ac = vec2ind(a)
```

运行上述程序后，系统首先输出训练用样本及其类别分类图，如图 6-66 所示。接着输出 PNN 的训练结果图，如图 6-67 所示。

训练用样本及其类别分类图与 PNN 的训练结果图是完全吻合的，用户可改变视角查看。此外，通过训练后的 PNN 对训练数据进行分类后的结果与目标结果对比表也可验证上述结论，如表 6-10 所示。

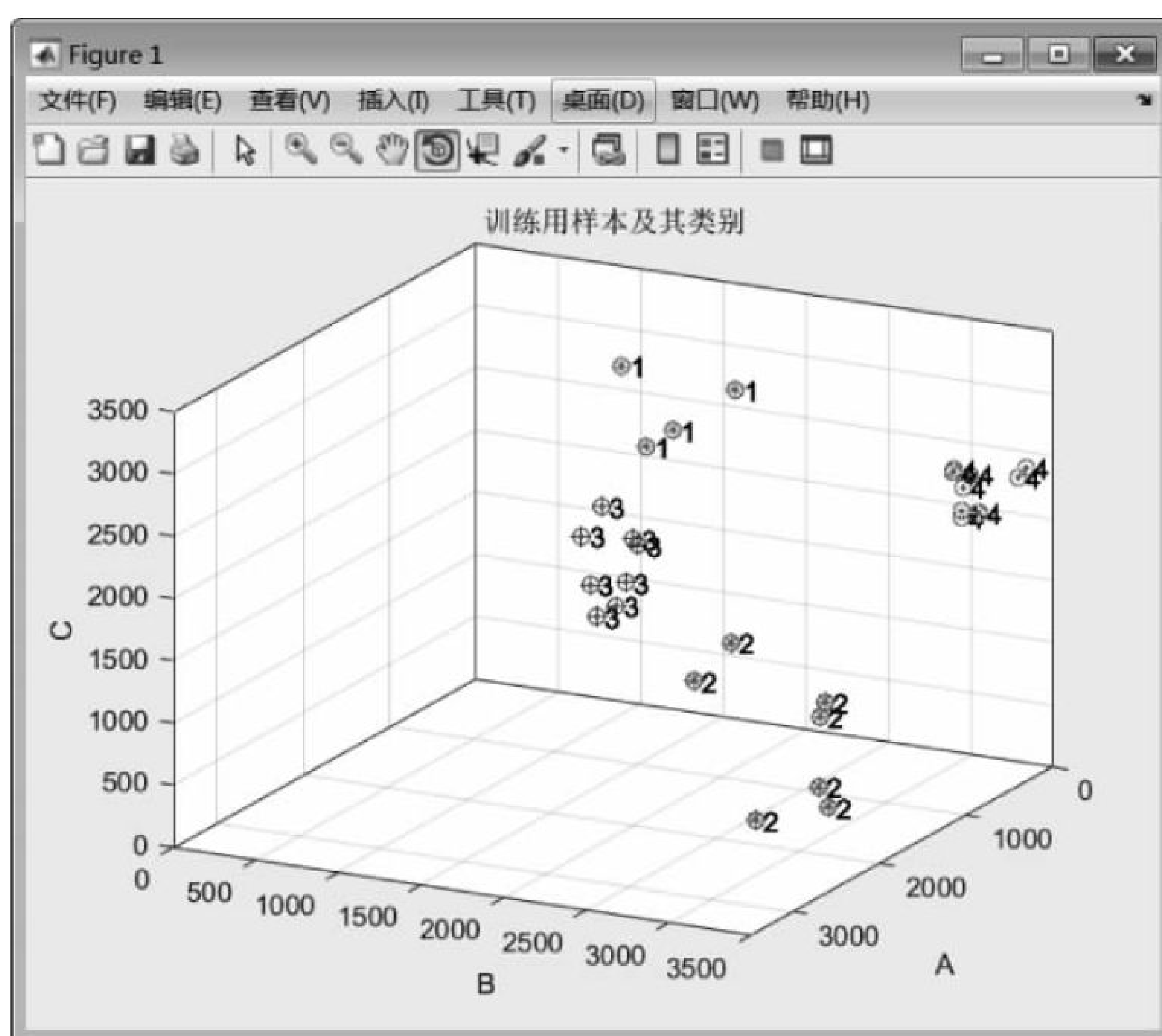


图 6-66 训练用样本及其类别分类图

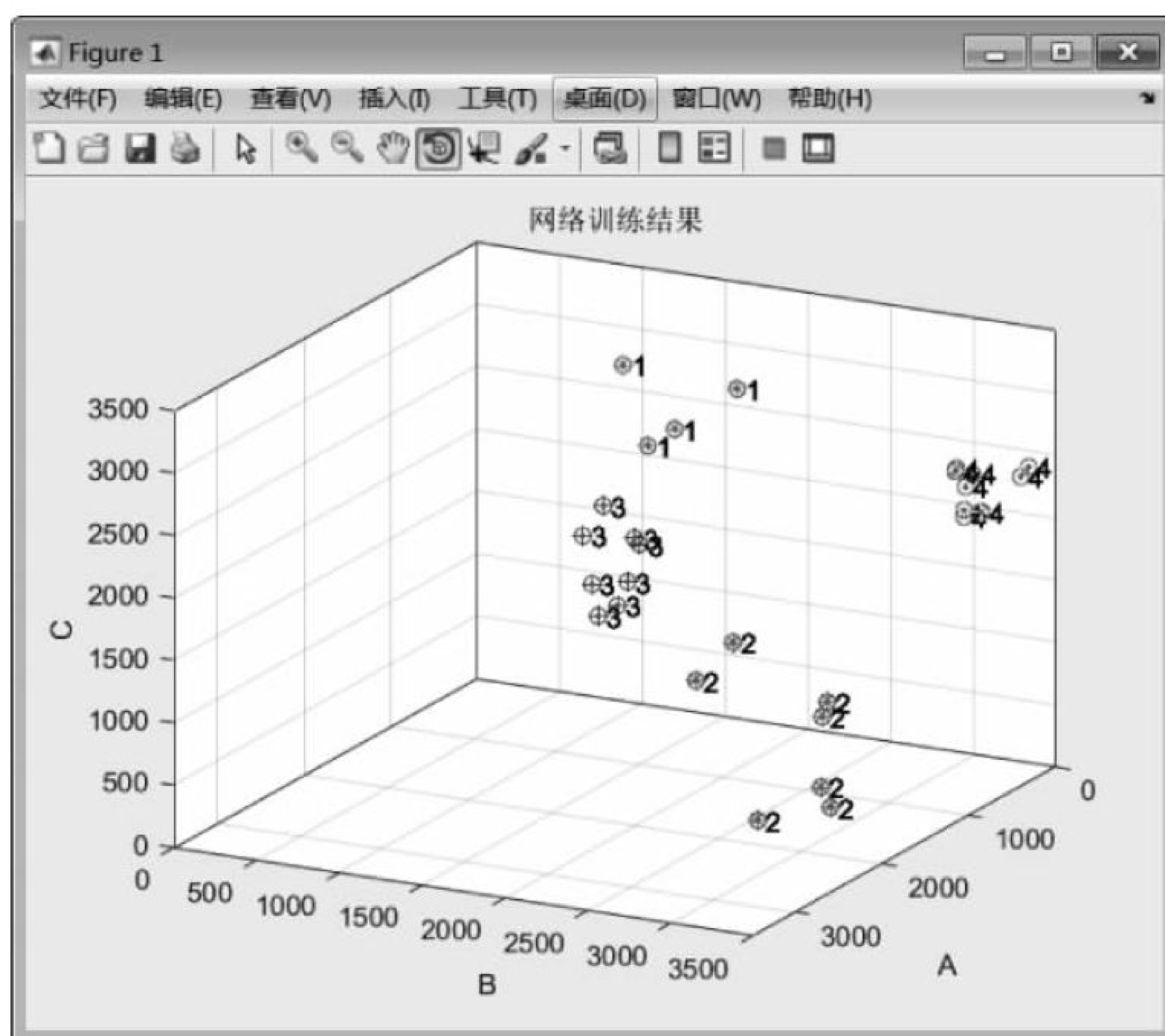


图 6-67 PNN 的训练结果图

表 6-10 训练后的 PNN 对训练数据进行分类后的结果与目标结果对比

序 号	A	B	C	目标结果	PNN 分类结果
4	864.45	1647.31	2665.9	1	1
6	877.88	2031.66	3071.18	1	1
16	1418.79	1775.89	2772.9	1	1
25	1449.58	1641.58	3405.12	1	1
8	2352.12	2557.04	1411.53	2	2
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
22	2802.88	3017.11	1984.98	2	2
24	2063.54	3199.76	1257.21	2	2
1	1739.94	1675.15	2395.96	3	3
3	1756.77	1652	1514.98	3	3
7	1803.58	1583.12	2163.05	3	3
11	1571.17	1731.04	1735.33	3	3
17	1845.59	1918.81	2226.49	3	3
20	1692.62	1867.5	2108.97	3	3
21	1680.67	1575.78	1725.1	3	3
26	1651.52	1713.28	1570.38	3	3
2	373.3	3087.05	2429.47	4	4
5	222.85	3059.54	2002.33	4	4
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
23	172.78	3084.49	2328.65	4	4
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4

训练后的 PNN 对训练数据进行分类后的结果与目标结果完全吻合，可见 PNN 训练效果良好。

继续执行程序，可得到待分类样本的分类结果：

```
ac =
1 ～ 15 列
1    1    1    1    2    2    2    2    2    2    2    3    3    3
3
16 ～ 30 列
3    3    3    3    4    4    4    4    4    4    4    4    4    4
3
31 ～ 45 列
1    3    4    2    2    3    4    1    3    3    1    2    4    2
```

```
4
46 ～ 48 列
3    4    2
```

4. RBF 与 PNN 神经网络分类器比较

RBF 神经网络对待分类样本的分类结果如下：

```
a =
1 ～ 9 列
1.0000  1.0000  1.0000  1.0000  2.0000  2.0000  2.0000  2.0000  2.0000
10 ～ 18 列
2.0000  2.0000  3.0000  3.0000  3.0000  3.0000  3.0000  3.0000  3.0000
19 ～ 27 列
3.0000  4.0000  4.0000  4.0000  4.0000  4.0000  4.0000  4.0000  4.0000
28 ～ 36 列
4.0000  4.0000  2.8969  3.2124  2.9232  4.0000  2.2147  2.4485  3.0550
37 ～ 45 列
4.0009  2.6013  3.1286  3.1476  1.3241  2.1283  4.0008  3.6605  3.9999
46 ～ 48 列
3.1801  3.9996  1.8306
a 为测试数据的分类结果,对 a 进行近似处理后可得最终的分类结果:
a =
1 ～ 9 列
1       1       1       1       2       2       2       2       2
10 ～ 18 列
2       2       3       3       3       3       3       3       3
19 ～ 27 列
3       4       4       4       4       4       4       4       4
28 ～ 36 列
4       4       3       3       3       4       2       2.4485 3
37 ～ 45 列
4       2.6013 3       3       1       2       4       3.6605 4
46 ～ 48 列
3       4       2
```

PNN 神经网络对待分类样本的分类结果如下：

```
ac =
1 ～ 15 列
1    1    1    1    2    2    2    2    2    2    2    3    3    3
3
16 ～ 30 列
3    3    3    3    4    4    4    4    4    4    4    4    4    4
3
31 ～ 45 列
1    3    4    2    2    3    4    1    3    3    1    2    4    2
4
46 ～ 48 列
3    4    2
```

从上面两组数据可以看出，对于径向基函数神经网络的分类结果 a，那些不能确定具体类别的数据，概率神经网络却给出了明确的分类结果。另外，对于同一组数据两种分类器得出了不同的结果，但用概率神经网络得到的分类结果与人工分类结果相同，这说明在用径向基函数神经网络分类器进行分类时，由于函数本身在数据分类方面的缺陷以及人为调整散布常数等问题，造成了最终分类结果的不准确。

径向基神经网络和概率神经网络均可以实现对给定数据的分类，并且概率神经网络的分类结果更理想。这是因为径向基网络主要用于函数的逼近，而概率神经网络主要用于解决分类问题，可以弥补径向基网络在分类方面的不足。

6.8.5 CPN 神经网络分类器的 MATLAB 实现

1. CPN 的结构

对向传播网络（counter propagation net，CPN），是将 Kohonen 特征映射网络与 Grossberg 基本竞争型网络相结合，发挥各自特长的一种新型特征映射网络。它是美国计算机专家 Robert Hecht-Nielsen 于 1987 年提出的。这种网络被广泛地用于模式分类、函数近似、统计分析和数据压缩等领域。CPN 结构如图 6-68 所示。

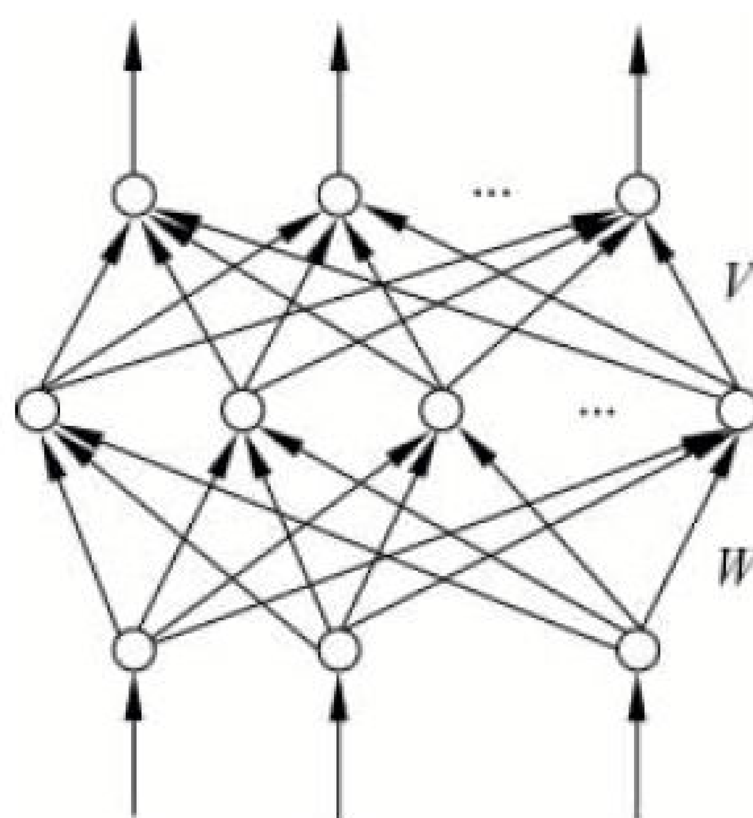


图 6-68 CPN 的结构

CPN 分为输入层、竞争层和输出层。输入层与竞争层构成 SOM 网络，竞争层与输出层构成基本竞争型网络。从整体上看，网络属于有教师型的网络，而由输入层和竞争层构成的 SOM 网络又是一种典型的无教师型神经网络。因此，这一网络既汲取了无教师型网络分类灵活、算法简练的优点，又采纳了有教师型网络分类精细、准确的长处，使两种不同类型的网络有机地结合起来。

CPN 的基本思想是，由输入层至输出层，网络按照 SOM 学习规则产生竞争层的获胜神经元，并按这一规则调整相应的输入层至竞争层的连接权；由竞争层到输出层，网络按照基本竞争型网络学习规则，得到各输出神经元的实际输出值，并按照有教师型的误差校正方法，修正由竞争层到输出层的连接权。经过这样的反复学习，可以将任意的输入模式映射为输出模式。

从这一基本思想可以发现，处于网络中间位置的竞争层获胜神经元使用与其相关的连接权向量，既反映了输入模式的统计特性，又反映了输出模式的统计特性。因此，可以认为，输入、输出模式通过竞争层实现了相互映射，即网络具有双向记忆的功能。如果输入/输出采用相同的模式对网络进行训练，则由输入模式至竞争层的映射可以认为是对输入模式的压缩；而由竞争至输出层的映射可以认为是对输入模式的复原。利用这一特性，可以有效地解决图像处理及通信中的数据压缩及复原问题，并可得到较高的压缩比。

2. CPN 的学习及工作规则

假定输入层有 N 个神经元，p 个连续值的输入模式为 $\boldsymbol{A}_k=(a_1^k,a_2^k,\cdots,a_N^k)$，竞争层有

Q个神经元，对应的二值输出向量为$\boldsymbol{B}_k=(b_1^k,b_2^k,\cdots,b_Q^k)$，输出层有$M$个神经元，其连续值的输出向量为$\boldsymbol{C}_k'=(c_1'^k,c_2'^k,\cdots,c_M'^k)$，目标输出向量为$\boldsymbol{C}_k=(c_1^k,c_2^k,\cdots,c_M^k)$。其中，$k=1,2,\cdots,p$。

由输入层至竞争层的连接权向量为$\boldsymbol{W}_j=(\omega_{j1},\omega_{j2},\cdots,\omega_{jN})$，$j=1,2,\cdots,Q$；由竞争层到输出层的连接权向量为$\boldsymbol{V}_l=(v_{l1},v_{l2},\cdots,v_{lQ})$，$l=1,2,\cdots,M$。

(1) 初始化。将连接权向量$\boldsymbol{W}_j$和$\boldsymbol{V}_l$赋予区间[0,1]内的随机值。将所有的输入模式A_k进行归一化处理：$a_1^k=\dfrac{a_1^k}{\|A_k\|}$，其中$\|A_k\|=\sqrt{\sum_{i=1}^{N}(a_i^k)^2}$，$k=1,2,\cdots,p$。

(2) 将第k个输入模式A_k提供给网络的输入层。

(3) 将连接权向量$\boldsymbol{W}_j$按照下式进行归一化处理：$\omega_{j1}=\dfrac{\omega_{j1}}{\|\omega_{j1}\|}$，其中$\|\omega_{j1}\|=\sqrt{\sum_{i=1}^{N}\omega_{j1}^2}$，$j=1,2,\cdots,Q$。

(4) 求竞争层中每个神经加权输入和：$s_j=\sum_{j=1}^{Q}a_i^k\omega_{ji}$，其中$i=1,2,\cdots,N$。

(5) 求连接权向量$\boldsymbol{W}_j$中与A_k距离最近的向量$\boldsymbol{W}_g$：

$\boldsymbol{W}_g=\max\limits_{j=1,2,\cdots,Q}\sum_{i=1}^{N}a_j^k\omega_{ji}=\max\limits_{j=1,2,\cdots,Q}s_j$，将神经元$g$的输出设定为1，其余竞争层神经元的输出设定为0，即$b_j=\begin{cases}1, & j=g\\0, & j\neq g\end{cases}$。

(6) 将连接权向量$\boldsymbol{W}_g$按照下式进行修正：

$$\omega_{gi}(t+1)=\omega_{gi}(t)+\alpha[a_i^k-\omega_{gi}(t)] \tag{6-94}$$

其中，$i=1,2,\cdots,N$；$-1<\alpha<1$，为学习率。

(7) 将连接权向量$\boldsymbol{W}_g$重新归一化，归一化算法同上。

(8) 按照下式修正竞争层到输出层的连接权向量$\boldsymbol{V}_l$：

$$v_{li}(t+1)=v_{li}(t)+\beta b_j(c_l-c_l') \tag{6-95}$$

其中，$l=1,2,\cdots,M$；$i=1,2,\cdots,N$；$j=1,2,\cdots,Q$；β为学习率。由步骤(5)可将上式简化为

$$v_{lg}(t+1)=v_{lg}(t)+\beta b_j(c_l-c_l') \tag{6-96}$$

由此可见，只需要调整竞争层获胜神经元g到输出层神经元的连接权向量$\boldsymbol{V}_g$即可，其他连接权向量保持不变。

(9) 求输出层各神经元的加权输入，并将其作为输出神经元的实际输出值，$c'_l=\sum_{j=0}^{Q}b_jv_{lg}$，其中$l=1,2,\cdots,M$。同理可将其简化为$c_l'=v_{lg}$。

(10) 返回步骤(2)，直到将p个输入模式全部提供给网络。

(11) 令$t=t+1$，将输入模式A_k重新提供给网络学习，直到$t=T$。其中，T为预先设定的学习总次数，一般取$500<T<10000$。

3. CPN神经网络用于模式分类

三元色数据如表1-1所示。其中，前29组数据已确定类别，后30组数据待确定类别。

其 CPN 神经网络分类 MATLAB 程序代码如下：

```
%初始化正向权值 w 和反向权值 v
w = rands(13,3)/2 + 0.5;
v = rands(1,13)/2 + 0.5;
%输入向量 P 和目标向量 T
P = importdata('C:\Users\Administrator\Desktop\ln\SelfOrganizationtrain.dat');
T = importdata('C:\Users\Administrator\Desktop\ln\SelfOrganizationtarget.dat');
T_out = T;
%设定学习步数为 1000 次
epoch = 1000;
%归一化输入向量 P
for i = 1:29
    if P(i,:) == [0 0 0]
            P(i,:) = P(i,:);
    else
            P(i,:) = P(i,:)/norm(P(i,:));
    end
end
%开始训练
while epoch > 0
for j = 1:29
    %归一化正向权值 w
        for i = 1:13
            w(i,:) = w(i,:)/norm(w(i,:));
            s(i) = P(j,:) * w(i,:)';
        end
    %求输出为最大的神经元,即获胜神经元
      temp = max(s);
      for i = 1:13
          if temp == s(i)
                count = i;
          end
      end
    % 将所有竞争层神经单元的输出置 0
      for i = 1:13
          s(i) = 0;
      end
    % 将获胜的神经元输出置 1
      s(count) = 1;
    %权值调整
      w(count,:) = w(count,:) + 0.1 * [P(j,:) - w(count,:)];
      w(count,:) = w(count,:)/norm(w(count,:));
      v(:,count) = v(:,count) + 0.1 * (T(j,:)' - T_out(j,:)');
    %计算网络输出
      T_out(j,:) = v(:,count)';
end
%训练次数递减
epoch = epoch - 1;
end
```

```
%训练结束
T_out
%网络回想
%网络的输入模式 Pc
Pc = importdata('C:\Users\Administrator\Desktop\ln\SelfOrganizationSimulation.dat');
%初始化 Pc
for i = 1:20
    if Pc(i,:) == [0 0 0]
          Pc(i,:) = Pc(i,:);
    else
          Pc(i,:) = Pc(i,:)/norm(Pc(i,:));
    end
end
%网络输出
Outc = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;];
for j = 1:20
    for i = 1:13
        sc(i) = Pc(j,:) * w(i,:)';
    end
        tempc = max(sc);
for i = 1:13
    if tempc == sc(i)
        countp = i;
    end
        sc(i) = 0;
end
        sc(countp) = 1;
        Outc(j,:) = v(:,countp)';
end
%回想结束
Outc
```

运行上述程序,系统给出使用 CPN 神经网络训练的分类器对样本数据的分类结果如下:

```
T_out =
    3.0000
    4.0000
    3.0000
    1.0000
    4.0000
    1.0000
    3.0000
    2.0000
    4.0000
    4.0000
    3.0000
    4.0000
    4.0000
    2.0000
    2.0000
    1.0000
```

```
3.0000
2.0000
2.0000
3.0000
3.0000
2.0000
4.0000
2.0000
1.0000
3.0000
4.0000
4.0000
4.0000
```

将系统的输出结果与目标结果对比，结果如表 6-11 所示。

表 6-11 训练后的 CPN 对训练数据进行分类后的结果与目标结果对比

序 号	A	B	C	原始分类结果	CPN 分类结果
1	1739.94	1675.15	2395.96	3	3
2	373.3	3087.05	2429.47	4	4
3	1756.77	1652	1514.98	3	3
4	864.45	1647.31	2665.9	1	1
5	222.85	3059.54	2002.33	4	4
6	877.88	2031.66	3071.18	1	1
7	1803.58	1583.12	2163.05	3	3
8	2352.12	2557.04	1411.53	2	2
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
11	1571.17	1731.04	1735.33	3	3
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
16	1418.79	1775.89	2772.9	1	1
17	1845.59	1918.81	2226.49	3	3
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
20	1692.62	1867.5	2108.97	3	3
21	1680.67	1575.78	1725.1	3	3
22	2802.88	3017.11	1984.98	2	2
23	172.78	3084.49	2328.65	4	4
24	2063.54	3199.76	1257.21	2	2
25	1449.58	1641.58	3405.12	1	1
26	1651.52	1713.28	1570.38	3	3
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4

训练后的 CPN 对训练数据进行分类后的结果与目标结果完全吻合。

继续运行程序,系统给出训练后的 CPN 对待分类样本的分类结果如下:

```
Outc =
    3.0000
    3.0000
    1.0000
    3.0000
    4.0000
    2.0000
    2.0000
    3.0000
    4.0000
    3.0000
    3.0000
    3.0000
    1.0000
    2.0000
    4.0000
    2.0000
    4.0000
    3.0000
    4.0000
    2.0000
```

多次运行程序,程序对待分类样本数据给出分类结果,如表 6-12 所示。

表 6-12 多次运行 CPN 对待分类样本数据的分类结果

序 号	次 数							
	1	2	3	4	5	6	7	8
1	3	3	3	3	3	3	3	3
2	3	3	3	3	3	3	3	3
3	1	0.9726	1	1	1	0.8664	0.1947	0.8020
4	3	2.5023	3	2.5023	2.7393	3	3	3
5	4	4	4	4	4	4	4	4
6	2	2	2	2	2	2	2	2
7	2	2	2	2	2	2	2	2
8	3	3	3	3	3	3	3	1.9524
9	4	4	4	4	4	4	4	4
10	0.9799	3	2.2689	3	3	1	3	1.9524
11	3	3	3	3	3	3	3	3
12	3	2.5023	3	2.5023	2.7393	3	3	3
13	1	1	1	1	1	1	1	1
14	2	2.5023	2	2.5023	2.7393	2	2	2
15	4	4	4	4	4	4	4	4
16	2	2	2	2	2	2	2	2

续表

序　号	次　　数							
	1	2	3	4	5	6	7	8
17	4	4	4	4	4	4	4	4
18	3	3	3	3	3	3	3	3
19	4	4	4	4	4	4	4	4
20	2	2	2	2	2	2	2	2
21	2	2	2	2	2	2	2	2
22	3	3	2.2689	3	3	3	3	1.9524
23	3	3	3	0.6795	3	3	3	3
24	1	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1	1
26	4	4	4	4	4	4	4	4
27	1	1	1	1	1	1	1	1.9524
28	3	3	3	3	3	3	3	3
29	3	3	3	3	3	3	3	3
30	3	3	3	3	3	3	3	3

序　号	次　　数							
	9	10	11	12	13	14	15	16
1	3	3	3	3	3	3	3	3
2	3	3	3	3	3	3	3	3
3	0.0419	1	0.3218	1	0.9597	1	0.0305	0.1711
4	3	3	3	3	3	3	3	3
5	3	4	4	4	4	4	4	4
6	3	2	2	2	2	2	2	2
7	3	2	2	2	2	2	2	2
8	3	3	3	3	3	3	3	3
9	3	4	4	4	4	4	4	4
10	1	3	0.3218	3	3	1.6042	1	3
11	3	3	3	3	3	3	3	3
12	3	3	3	3	3	3	3	3
13	1	1	1	0.9876	1	1	1	1
14	2	2	2	2	2	2	2	2
15	4	4	4	4	4	4	4	4
16	2	2	2	2	2	2	2	2
17	4	4	4	4	4	4	4	4
18	3	3	3	3	3	3	3	3
19	4	1.1603	4	4	4	4	4	4
20	2	2	2	2	2	2	2	2
21	2	2	2	2	2	2	2	2
22	1	3	1	3	3	1.6042	3	3
23	3	3	3	3	3	3	3	3
24	1	1	1	1	1	1.6042	0.7150	1
25	1	1	1	1	1	1	1	1
26	4	4	4	4	4	4	4	4
27	1	1	1	1	1	1.6042	0.7150	1
28	3	3	3	3	3	3	3	3
29	3	3	3	3	3	3	3	3
30	3	3	3	3	3	3	3	3

从表 6-12 中可以看出,用 CPN 神经网络出现数据不稳定主要是由于 CPN 算法设计的不完善所致。但仔细观察序号为 10 的数据的 3 个特征值,特征值 A 为 1494.63,与第三类中的序号为 8 的数据的特征值 A(1507.13)极其相近,而且特征值 B 和 C 与第 3 类中样本的特征值也相差不远,这也是被 CPN 神经网络误判的一个原因。

总之,CPN 神经网络在模式分类上有较高的准确率,可以正确、有效、快速地区分不同的特征点,学习时间较快,学习效率较高。

习题

(1) 神经网络的发展可以分为几个阶段,各个阶段对神经网络的发展有何意义?

(2) 什么是人工神经网络?

(3) 请对照神经细胞的工作机理,分析人工神经元基本模型的工作原理。

(4) 简述人工神经网络的工作过程。

(5) 基于神经元构建的人工神经网络具有哪些特点?

(6) 感知器网络具有哪些特点?

(7) 简述 BP 网络学习算法的主要思想。

(8) 简述 BP 网络的建立方式及执行过程。

(9) 简述离散 Hopfield 网络的工作方式及连接权设计的主要思想。

(10) 简述径向基函数网络的工作方式、特点、作用及参数选择的方法。

(11) 从资料中得到某地区的 12 个风蚀数据,每个样本数据用 6 个指标表示其性状,请分别用 BP 网络、离散 Hopfield 网络、RBF 网络、自组织竞争网络、SOM 网络、LVQ 网络、PNN 及 CPN 设计模式分类系统。

原始数据如表 6-13 所示。

表 6-13 某地区风蚀样本数据及其性状

序号	风蚀危险度	土壤细沙含量/g	沙地面积所占比例/%	地形起伏度/cm	风场强度/(km·s^{-1})	2—5 月 NDVI 平均值	土壤干燥度/%
1	轻度	0.41	0.00	0.75	0.07	0.67	0.01
2	轻度	0.41	0.00	0.62	0.14	0.67	0.01
3	中度	0.68	0.3	0.22	0.12	0.41	0.04
4	中度	0.5	0.51	0.01	0.28	0.55	0.04
5	强度	0.80	0.96	0.15	0.05	0.17	0.10
6	强度	0.72	0.93	0.11	0.87	0.18	0.11
7	极强	0.62	0.91	0.29	0.29	0.05	0.44
8	极强	0.47	0.79	0.13	0.71	0.00	1.00
9	轻度	0.52	0.00	0.66	0.12	0.75	0.02
10	中度	0.69	0.52	0.57	0.12	0.54	0.04
11	强度	0.63	0.69	0.22	0.19	0.12	0.11
12	极强	0.49	0.86	0.18	0.23	0.07	0.25

注:NDVI 为归一化植被指数,无单位。

要求使用前 8 组数据作为训练数据,后 4 组数据作为测试数据。

第7章

模拟退火算法聚类设计

作为计算智能方法的一种，模拟退火方法于20世纪80年代初被提出，其基本的思想来源于固体的退火过程。1982年，Kirkpatrick等人首先意识到固体退火过程与优化问题之间存在着类似性，Metropolis等人对固体在恒定温度下达到热平衡过程的模拟也给他们以启迪。通过把Metropolis准则引入到优化过程中来，最终他们得到一种对Metropolis算法进行迭代的优化算法，这种算法类似固体退火过程，故称为“模拟退火算法”。

7.1 模拟退火算法简介

模拟退火算法是一种适合求解大规模组合优化问题的随机搜索算法。目前，模拟退火算法在求解TSP、VLSI电路设计等组合优化问题上取得了令人满意的结果。由于模拟退火算法具有适用范围广、求得全局最优解的可靠性高、算法简单、便于实现等优点，在应用于求解连续变量函数的全局优化问题上得到了广泛的研究，同时取得了很好的效果。但是，为了提高模拟退火算法的搜索效率，国内外的科研人员还在进行各种研究工作，并尝试从模拟退火算法的不同阶段入手对算法进行改进。同时，将模拟退火算法同其他的计算智能方法相结合，应用到各类复杂系统的建模和优化问题中也得到了越来越多的重视，已经逐渐成为一个重要的发展方向。

7.1.1 物理退火过程

模拟退火算法得益于材料的统计力学的研究成果。统计力学表明材料中粒子的不同结构对应于粒子的不同能量水平。在高温条件下，粒子的能量较高，可以自由运动和重新排序。在低温条件下，粒子能量较低。物理退火过程如图7-1所示，整个过程由三部分组成。

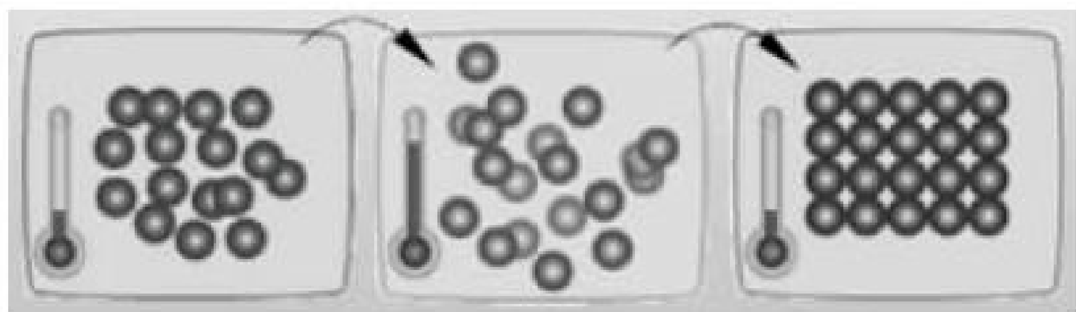


图7-1 物理退火过程

1. 升温过程

升温的目的是增强物体中粒子的热运动,使其偏离平衡位置成为无序状态。当温度足够高时,固体将溶解为液体,从而消除系统原先可能存在的非均匀态,使随后的冷却过程以某一平衡态为起点。升温过程与系统的熵增过程相关,系统能量随温度的升高而增大。

2. 等温过程

在物理学中,对于与周围环境交换热量而温度不变的封闭系统,系统状态的自发变化总是朝向自由能减小的方向进行。当自由能达到最小时,系统达到平衡态。

3. 冷却过程

与升温过程相反,使物体中粒子的热运动减弱并渐趋有序,系统能量随温度降低而下降,得到低能量的晶体结构。

7.1.2 Metropolis 准则

固体在恒定的温度下达到热平衡的过程可以用 Monte Carlo 方法进行模拟。Monte Carlo 方法的特点是算法简单,但必须大量采样才能得到比较精确的结果,因而计算量很大。

1953 年,Metropolis 等人提出了一种重要的采样法。他们用下述方法产生固体的状态序列:

先给定以粒子相对位置表征的初始状态 i,作为固体的当前状态,该状态的能量是 E_i。然后用摄动装置使随机选取的某个粒子的位移随机地产生一微小变化,得到一个新状态 j,新状态的能量是 E_j。如果 $E_j \leqslant E_i$,则该新状态就作为“重要”状态;如果 $E_j > E_i$,则要考虑到热运动的影响。该新状态是否为“重要”状态,要依据固体处于该状态的概率,由 $p = \exp\left(\frac{E_i - E_j}{kT}\right)$ 式来判断,式中 k 为物理学中的玻耳兹曼常数;T 为材料的热力学温度。

p 是一个小于 1 的数。用随机数发生器产生一个[0,1)区间的随机数 ξ,若 $p > \xi$,则新状态 j 作为重要状态,否则舍弃。若新状态 j 是重要状态,就以 j 取代 i 成为当前状态,否则仍以 i 为当前状态。再重复以上新状态的产生过程。

由 $p = \exp\left(\frac{E_i - E_j}{kT}\right)$ 可知,高温下可接受与当前状态能差较大的新状态为重要状态,而在低温下只能接受与当前状态能差较小的新状态为重要状态。这与不同温度下热运动的影响完全一致。在温度趋于 0℃时,就不能接受任一 $E_j > E_i$ 的新状态 j 了。

7.1.3 模拟退火算法的基本原理

模拟退火算法来源于固体退火原理,将固体加温至充分高,再让其徐徐冷却。加温时,固体内部粒子随温升变为无序状,内能增大,而徐徐冷却时粒子渐趋有序,在每个温度都达到平衡态,最后在常温时达到基态,内能减为最小。

根据 Metropolis 准则,粒子在温度 T 时趋于平衡的概率为 $\exp[-\Delta E/(kT)]$。其中,

E 为温度 T 时的内能；ΔE 为其改变量；k 为玻耳兹曼常数。用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法：由初始解 i 和控制参数初值 t 开始，对当前解重复"产生新解→计算目标函数差→接受或舍弃"的迭代，并逐步衰减 t 值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。

退火过程由冷却进度表(cooling schedule)控制，包括控制参数的初值 t 及其衰减因子 Δt、每个 t 值时的迭代次数 L 和停止条件 S。

模拟退火与组合优化算法具有相似性，如表 7-1 所示。

表 7-1　组合优化与模拟退火的相似性

组 合 优 化	模 拟 退 火
解	粒子状态
最优解	能量最低态
目标函数	能量
设定温度	溶解工程
Metropolis 抽样过程	等温过程
控制参数的下降	冷却

7.1.4　模拟退火算法的组成

模拟退火算法由解空间、目标函数和初始解组成。

(1) 解空间：对所有可能解均为可行解的问题定义为可能解的集合，对存在不可行解的问题，或限定解空间为所有可行解的集合，或允许包含不可行解但在目标函数中用罚函数惩罚以致最终完全排除不可行解。

(2) 目标函数：对优化目标的量化描述，是解空间到某个数集的一个映射，通常表示为若干优化目标的一个和式，应正确体现问题的整体优化要求且较易计算。当解空间包含不可行解时还应包括罚函数项。

(3) 初始解：是算法迭代的起点。试验表明，模拟退火算法是健壮的，即最终解的求得不十分依赖初始解的选取，从而可任意选取一个初始解。

7.1.5　模拟退火算法新解的产生和接受

模拟退火算法新解的产生和接受可分为以下四个步骤。

(1) 由一个产生函数从当前解产生一个位于解空间的新解。为便于后续的计算和接受，减少算法耗时。通常选择由当前新解经过简单变换即可产生新解的方法，如对构成新解的全部或部分元素进行置换、互换等。产生新解的变换方法决定了当前新解的邻域结构，因而对冷却进度表的选取有一定的影响。

(2) 计算与新解所对应的目标函数差。因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。事实表明，对大多数应用而言，这是计算目标函数差的最快方法。

(3) 判断新解是否被接受。判断的依据是一个接受准则，最常用的接受准则是 Metropolis 准则：若 $\Delta t'<0$，则接受 S' 作为新的当前解 S；否则，以概率 $\exp(-\Delta t'/T)$ 接受 S' 作为新的当前解 S。

(4) 当新解被确定接受时，用新解代替当前解，这只需将当前解中对应于产生新解时的变换部分予以实现，同时修正目标函数值即可。此时，当前解实现了一次迭代。可在此基础上开始下一轮试验。当新解被判定为舍弃时，则在原当前解的基础上继续下一轮试验。

7.1.6 模拟退火算法的基本过程

(1) 初始化，给定初始温度 T_0 及初始解 w，计算解对应的目标函数值 $f(w)$。在本节中 w 代表一种聚类划分。

(2) 模型扰动产生新解 w' 及对应的目标函数值 $f(w')$。

(3) 计算函数差值 $\Delta f = f(w') - f(w)$。

(4) 如果 $\Delta f \leqslant 0$，则接受新解作为当前解。

(5) 如果 $\Delta f > 0$，则以概率 p 接受新解。

$$p = \mathrm{e}^{-[f(w')-f(w)]/f(KT)} \tag{7-1}$$

(6) 对当前 T 值降温，对步骤(2)～(5)迭代 N 次。

(7) 如果满足终止条件，输出当前解为最优解，结束算法；否则，降低温度，继续迭代。

模拟退火算法的流程如图 7-2 所示。算法中包含一个内循环和一个外循环。内循环就是在同一温度下的多次扰动产生不同模型状态，并按照 Metropolis 准则接受新模型，因此是用模型扰动次数控制的；外循环包括了温度下降的模拟退火算法的迭代次数的递增和算法停止的条件，因此基本是用迭代次数控制的。

7.1.7 模拟退火算法的参数控制问题

模拟退火算法的应用很广泛，可以求解 NP 完全问题，但其参数难以控制，主要存在以下三个问题：

1. 温度 *T* 的初始值设置问题

温度 T 的初始值设置是影响模拟退火算法全局搜索性能的重要因素之一。初始温度高，则搜索到全局最优解的可能性大，但因此要花费大量的计算时间；反之，则可节省计算时间，但全局搜索性能可能受到影响。实际应用过程中，初始温度一般需要依据实验结果进行若干次调整。

2. 退火速度问题

模拟退火算法的全局搜索性能也与退火速度密切相关。一般来说，同一温度下的“充分”搜索(退火)是相当必要的，但这需要计算时间。实际应用中，要针对具体问题的性质和特征设置合理的退火平衡条件。

3. 温度管理问题

温度管理问题也是模拟退火算法难以处理的问题之一。实际应用中，由于必须考虑计

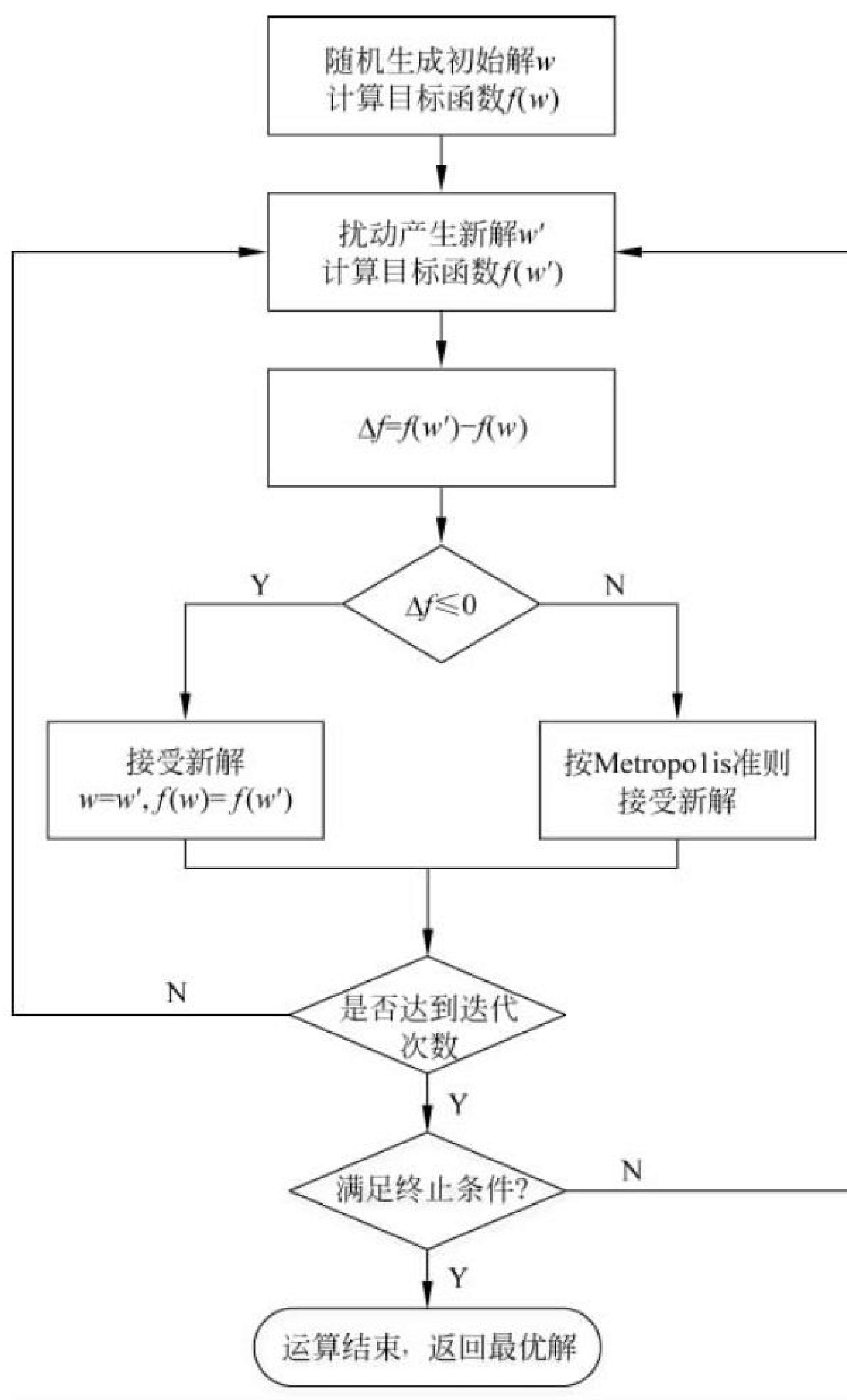


图 7-2　模拟退火算法流程图

算复杂度的切实可行性等问题，常采用降温方式 $T(t+1)=k\times T(t)$。式中，k 为正的略小于 1.00 的常数；t 为降温的次数。

7.2　基于模拟退火思想的聚类算法

7.2.1　K 均值算法的局限性

基本的 K 均值算法目的是找到使目标函数值最小的 K 个划分，算法思想简单，易实现，而且收敛速度较快。如果各个簇之间区别明显，且数据分布稠密，则该算法比较有效，但如果各个簇的形状和大小差别不大，则可能会出现较大的簇分割现象。此外，在 K 均值算法聚类时，最佳聚类结果通常对应于目标函数的极值点，由于目标函数可能存在很多的局部极小值点，这就会导致算法在局部极小值点处收敛。因此，初始聚类中心的随机选取可能会

使解陷入局部最优解，难以获得全局最优解。

该算法的局限性主要表现在以下方面。

(1) 最终的聚类结果依赖于最初的划分。

(2) 需要事先指定聚类的数目 M。

(3) 产生的类大小相关较大，对于噪声和孤立点敏感。

(4) 算法经常陷入局部最优。

(5) 不适合对非凸面形状的簇或差别很小的簇进行聚类。

7.2.2 基于模拟退火思想的改进 K 均值聚类算法

模拟退火算法是一种启发式随机搜索算法，具有并行性和渐近收敛性，已在理论上证明它是一种以概率为 1，收敛于全局最优解的全局优化算法，因此用模拟退火算法对 K 均值聚类算法进行优化，可以改进 K 均值聚类算法的局限性，提高算法性能。

基于模拟退火思想的改进 K 均值聚类算法中，将内能 E 模拟为目标函数值，将基本 K 均值聚类算法的聚类结果作为初始解，初始目标函数值作为初始温度 T_0，对当前解重复“产生新解→计算目标函数差→接受或舍弃新解”的迭代过程，并逐步降低 T 值，算法终止时当前解为近似最优解。这种算法开始时以较快的速度找到相对较优的区域，然后进行更精确的搜索，最终找到全局最优解。

7.2.3 几个重要参数的选择

1. 目标函数

选择当前聚类划分的总类间离散度作为目标函数

$$J_w = \sum_{i=1}^{M} \sum_{\boldsymbol{X} \in w_i} d(\boldsymbol{X}, \overline{X^{(w_i)}}) \tag{7-2}$$

式中，$\boldsymbol{X}$ 为样本向量；w 为聚类划分；$\overline{X^{(w_i)}}$ 为第 i 个聚类的中心；$d(\boldsymbol{X}, \overline{X^{(w_i)}})$ 为样本到对应聚类中心的距离；聚类准则函数 J_w 即为各类样本到对应聚类中心距离的总和。

2. 初始温度

一般情况下，为了使最初产生的新解被接受，在算法开始时就应达到准平衡，因此选取初始温度聚类结果 $T_0 = J_w$ 作为初始解。

3. 扰动方法

模拟退火算法中的新解的产生是对当前解进行扰动得到的。本算法采用一种随机扰动方法，即随机改变一个聚类样本的当前所属类别，从而产生一种新的聚类划分，使算法有可能跳出局部极小值。

4. 退火方式

模拟退火算法中，退火方式对算法有很大的影响。如果温度下降过慢，算法的收敛速度

会大大降低。如果温度下降过快,可能会丢失极值点。为了提高模拟退火算法的性能,许多学者提出了退火方式,比较有代表性的退火方式如下:(下面公式中 t 代表最外层当前循环次数,α 为可调参数,可以改善退火曲线的形态。)

$$T(t)=\frac{T_0}{\ln(1+t)} \tag{7-3}$$

其特点是温度下降缓慢,算法收敛速度也较慢。

$$T(t)=\frac{T_0}{\ln(1+\alpha t)} \tag{7-4}$$

α 为可调参数,可以改善退火曲线的形态。其特点是高温区温度下降较快,低温区温度下降较慢,即主要在低温区进行寻优。

$$T(t)=T_0\alpha^t \tag{7-5}$$

α 为可调参数,其特点是温度下降较快,算法收敛速度快。本算法采用此退火方式,其中 α 为退火速度,控制温度下降的快慢,取 $\alpha=0.99$。

7.3 算法实现

7.3.1 实现步骤

基于模拟退火思想的 K 均值聚类算法流程如图 7-3 所示。

如图 7-3 所示,算法中包含一个内循环和一个外循环。内循环就是在同一温度下的多次扰动产生不同模型状态,并按照 Metropolis 准则接受新模型,因此是用模型扰动次数控制的;外循环包括了温度下降的模拟退火算法的迭代次数的递增和算法停止的条件,因此基本是用迭代次数控制的。

(1) 对样本进行 K 均值聚类,将聚类划分结果作为初始解 w,根据 $J_w=\sum_{i=1}^{M}\sum_{\boldsymbol{X}\in w_i}d(\boldsymbol{X},\overline{X^{(w_i)}})$ 计算目标函数值 J_w。

(2) 初始化温度 T_0,令 $T_0=J_w$,初始化退火速度 α 和最大退火次数。

(3) 对于某一温度 t 在步骤(4)~(7)进行迭代,直到达到最大迭代次数,跳到步骤(8)。

(4) 随机扰动产生新的聚类划分 w',即随机改变一个聚类样本的当前所属类别,计算新的目标函数值 J'_w。

(5) 判断新的目标函数值 J'_w 是否为最优目标函数值,是则保存聚类划分 w' 为最优聚类划分、J'_w 为最优目标函数值;否则跳到下一步。

(6) 计算函数差值 $\Delta J=J'_w-J_w$。

(7) 判断 ΔJ 是否小于 0:

若 $\Delta J\leqslant 0$,则接受新解,即将新解作为当前解。

若 $\Delta J>0$,则根据 Metropolis 准则接受新解。

(8) 判断是否达到最大退火次数,是则结束算法,输出最优聚类划分;否则降低温度,继续迭代。

图 7-3　基于模拟退火思想的 K 均值聚类算法流程

7.3.2　模拟退火实现模式分类的 MATLAB 程序

1. 初始化程序

程序首先需要输入样本数目、待分类的数目、初始分类及其他的一些相关参数。初始化程序代码如下：

```
[num,n] = size(p);                  % 样本数目
centernum = 4;                      % 类别数目
IDX0 = [1 2 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 ];              %  设置样本的初始分类
time = 1;
Tbegin = 10;Tover = 0.1;            % 起始温度,终止温度
L = 500;                            % 内层循环次数
T = Tbegin;                         % 初始化温度参数
timeb = 0;                          % 最优目标首次出现的退火次数
```

2. 求初始聚类中心

其程序代码如下：

```
s4 = find(IDX0 == 4);                     % 聚类号为 4 的样本在 p 中的序号
s44 = p(s4,:);                            % 全部为 4 类的样本矩阵
C0(4,:) = [sum(s44(:,1))/48,sum(s44(:,2))/48,sum(s44(:,3))/48]; % 第 4 类的中心
J0 = 0;
j1 = 0; j2 = 0; j3 = 0; j4 = 0;
for i = 1:num
     if IDX0(i) == 4
         j4 = j4 + sqrt((p(i,1) - C0(1,1))^2 + (p(i,2) - C0(1,2))^2 + (p(i,3) - C0(1,3))^2);
     end
  end
    J0 = j1 + j2 + j3 + j4;                 % 4 种类别的类内所有点与该类中心的距离和
```

3. 产生随机扰动

产生随机扰动的程序代码如下：

```
%产生随机扰动,即随机改变一个聚类样本的当前所属类别
        t1 = fix(rand * num + 1);                    % 随机抽取一个样本
        t2 = fix(rand * (centernum - 1) + 1);        % 随机生成 1～3 的整数
        if(IDXN(t1) + t2 > centernum)
             IDXN(t1) = IDXN(t1) + t2 - centernum;
        else
             IDXN(t1) = IDXN(t1) + t2;
        end
```

4. 重新计算聚类中心

重新计算聚类中心的程序代码如下：

```
p1 = find(IDXN == 1);                                    % 聚类号为 1 的样本在 p 中的序号
p11 = p(p1,:);                                           % 全部为 1 类的样本矩阵
[b1,a1] = size(p1);
CN(1,:) = [sum(p11(:,1))/a1,sum(p11(:,2))/a1,sum(p11(:,3))/a1]; % 第一类的中心
p2 = find(IDXN == 2);                                    % 聚类号为 2 的样本在 p 中的序号
p22 = p(p2,:);                                           % 全部为 2 类的样本矩阵
[b2,a2] = size(p2);
CN(2,:) = [sum(p22(:,1))/a2,sum(p22(:,2))/a2,sum(p22(:,3))/a2]; % 第 2 类的中心
p3 = find(IDXN == 3);                                    % 聚类号为 3 的样本在 p 中的序号
p33 = p(p3,:);                                           % 全部为 3 类的样本矩阵
[b3,a3] = size(p3);
CN(3,:) = [sum(p33(:,1))/a3,sum(p33(:,2))/a3,sum(p33(:,3))/a3]; % 第 3 类的中心
p4 = find(IDXN == 4);                                    % 聚类号为 4 的样本在 p 中的序号
p44 = p(p4,:);                                           % 全部为 4 类的样本矩阵
```

```
[b4,a4] = size(p4);
CN(4,:) = [sum(p44(:,1))/a4,sum(p44(:,2))/a4,sum(p44(:,3))/a4]; % 第 4 类的中心
```

5. 计算目标函数

计算目标函数的程序代码如下：

```
JN = 0;
j1 = 0; j2 = 0; j3 = 0; j4 = 0;
for i = 1:num
    if IDXN(i) == 1
        j1 = j1 + sqrt((p(i,1) - CN(1,1))^2 + (p(i,2) - CN(1,2))^2 + (p(i,3) - CN(1,3))^2);
    elseif IDXN(i) == 2
        j2 = j2 + sqrt((p(i,1) - CN(2,1))^2  + (p(i,2) - CN(2,2))^2 + (p(i,3) - CN(2,3))^2);
    elseif IDXN(i) == 3
        j3 = j3 + sqrt((p(i,1) - CN(3,1))^2 + (p(i,2) - CN(3,2))^2 + (p(i,3) - CN(3,3))^2);
    elseif IDXN(i) == 4
        j4 = j4 + sqrt((p(i,1) - CN(4,1))^2 + (p(i,2) - CN(4,2))^2 + (p(i,3) - CN(4,3))^2);
    end
end
JN = j1 + j2 + j3 + j4;   % 四种类别的类内所有点与该类中心的距离和
e = JN - JO;
```

6. 判断是否接受新解

判断是否接受新解的程序代码如下：

```
if e <= 0
     JO = JN;CO = CN;IDXO = IDXN;
else
   if(rand < exp( - e/T))
          JO = JN;
          CO = CN;
          IDXO = IDXN;
   else
         IDXN = IDXO;IDX = IDXO;CN = CO;JN = JO;
   end
end
```

模拟退火实现模式分类的 MATLAB 完整程序代码如下：

```
% clc;
close all;clear all;
p = [1739.94       1675.15       2395.96
    373.3         3087.05       2429.47
    1756.77       1652          1514.98
    864.45        1647.31       2665.9
    222.85        3059.54       2002.33
```

```
877.88      2031.66     3071.18
1803.58     1583.12     2163.05
2352.12     2557.04     1411.53
401.3       3259.94     2150.98
363.34      3477.95     2462.86
1571.17     1731.04     1735.33
104.8       3389.83     2421.83
499.85      3305.75     2196.22
2297.28     3340.14     535.62
2092.62     3177.21     584.32
1418.79     1775.89     2772.9
1845.59     1918.81     2226.49
2205.36     3243.74     1202.69
2949.16     3244.44     662.42
1692.62     1867.5      2108.97
1680.67     1575.78     1725.1
2802.88     3017.11     1984.98
172.78      3084.49     2328.65
2063.54     3199.76     1257.21
1449.58     1641.58     3405.12
1651.52     1713.28     1570.38
341.59      3076.62     2438.63
291.02      3095.68     2088.95
237.63      3077.78     2251.96
1702.8      1639.79     2068.74
1877.93     1860.96     1975.3
867.81      2334.68     2535.1
1831.49     1713.11     1604.68
460.69      3274.77     2172.99
2374.98     3346.98     975.31
2271.89     3482.97     946.7
1783.64     1597.99     2261.31
198.83      3250.45     2445.08
1494.63     2072.59     2550.51
1597.03     1921.52     2126.76
1598.93     1921.08     1623.33
1243.13     1814.07     3441.07
2336.31     2640.26     1599.63
354         3300.12     2373.61
2144.47     2501.62     591.51
426.31      3105.29     2057.8
1507.13     1556.89     1954.51
343.07      3271.72     2036.94
2201.94     3196.22     935.53
2232.43     3077.87     1298.87
1580.1      1752.07     2463.04
1962.4      1594.97     1835.95
1495.18     1957.44     3498.02
1125.17     1594.39     2937.73
24.22       3447.31     2145.01
1269.07     1910.72     2701.97
1802.07     1725.81     1966.35
1817.36     1927.4      2328.79
1860.45     1782.88     1875.13
];
```

```
[num,n] = size(p);                                          % 样本数目
centernum = 4;                                              % 类别数目
IDXO = [1 2 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 ];
                                                            % size(IDXO)
CO(1,:) = [1739.94    1675.15    2395.96];
CO(2,:) = [373.3    3087.05    2429.47];
CO(3,:) = [1756.77    1652    1514.98];
 % s1 = find(IDXO == 1);                                    % 聚类号为 1 的样本在 p 中的序号
 % s11 = p(s1,:)
s4 = find(IDXO == 4);                                       % 聚类号为 4 的样本在 p 中的序号
s44 = p(s4,:);                                              % 全部为 4 类的样本矩阵
CO(4,:) = [sum(s44(:,1))/59,sum(s44(:,2))/59,sum(s44(:,3))/59]; % 第 4 类的中心
JO = 0;
j1 = 0; j2 = 0; j3 = 0; j4 = 0;
for i = 1:num
     if IDXO(i) == 4
          j4 = j4 + sqrt((p(i,1) - CO(1,1))^2 + (p(i,2) - CO(1,2))^2 + (p(i,3) - CO(1,3))^2);
     end
end
 JO = j1 + j2 + j3 + j4;                                    % 四种类别的类内所有点与该类中心的距离和
 JO
 C = CO;J = JO;IDX = IDXO;
time = 1;
Tbegin = 10;Tover = 0.1;                                    % 起始温度,终止温度
L = 300;                                                    % 内层循环次数
T = Tbegin;                                                 % 初始化温度参数
timeb = 0;                                                  % 最优目标首次出现的退火次数
 % K = 0.0001;
tic;
IDXN = IDXO;
while T > Tover
    tt = 0;
    for inner = 1:L
          % 产生随机扰动,即随机改变一个聚类样本的当前所属类别
         t1 = fix(rand * num + 1);                          % 随机抽取一个样本
         t2 = fix(rand * (centernum - 1) + 1);              % 随机生成 1～3 的整数
         if(IDXN(t1) + t2 > centernum)
              IDXN(t1) = IDXN(t1) + t2 - centernum;
         else
               IDXN(t1) = IDXN(t1) + t2;
         end
 %       t1 = fix(rand * (num - 1) + 1);                    % 随机抽取一个样本
 %       t2 = fix(rand * (centernum - 1) + 1);              % 随机生成 1～4 的整数
 %       if(IDXN(t1) + t2 > centernum)
 %            IDXN(t1) = IDXN(t1) + t2 - centernum;
 %       else
 %            IDXN(t1) = IDXN(t1) + t2;
 %       end
```

```
%      IDXN(t1) = t2;
       IDXN;
     %重新计算聚类中心
    p1 = find(IDXN == 1);                          %聚类号为 1 的样本在 p 中的序号
    p11 = p(p1,:);                                 %全部为 1 类的样本矩阵
    [b1,a1] = size(p1);
     CN(1,:) = [sum(p11(:,1))/a1,sum(p11(:,2))/a1,sum(p11(:,3))/a1];%第一类的中心
     p2 = find(IDXN == 2);                         %聚类号为 2 的样本在 p 中的序号
     p22 = p(p2,:);                                %全部为 2 类的样本矩阵
     [b2,a2] = size(p2);
     CN(2,:) = [sum(p22(:,1))/a2,sum(p22(:,2))/a2,sum(p22(:,3))/a2];%第 2 类的中心
     p3 = find(IDXN == 3);                         %聚类号为 3 的样本在 p 中的序号
     p33 = p(p3,:);                                %全部为 3 类的样本矩阵
     [b3,a3] = size(p3);
     CN(3,:) = [sum(p33(:,1))/a3,sum(p33(:,2))/a3,sum(p33(:,3))/a3];%第 3 类的中心
     p4 = find(IDXN == 4);                         %聚类号为 4 的样本在 p 中的序号
     p44 = p(p4,:);                                %全部为 4 类的样本矩阵
     [b4,a4] = size(p4);
     CN(4,:) = [sum(p44(:,1))/a4,sum(p44(:,2))/a4,sum(p44(:,3))/a4];%第 4 类的中心
      %计算目标函数
     JN = 0;
     j1 = 0; j2 = 0; j3 = 0; j4 = 0;
     for i = 1:num
          if IDXN(i) == 1
              j1 = j1 + sqrt((p(i,1) - CN(1,1))^2 + (p(i,2) - CN(1,2))^2 + (p(i,3) - CN(1,3))^2);
           elseif IDXN(i) == 2
              j2 = j2 + sqrt((p(i,1) - CN(2,1))^2  + (p(i,2) - CN(2,2))^2 + (p(i,3) - CN(2,3))^2);
           elseif IDXN(i) == 3
              j3 = j3 + sqrt((p(i,1) - CN(3,1))^2 + (p(i,2) - CN(3,2))^2 + (p(i,3) - CN(3,3))^2);
           elseif IDXN(i) == 4
              j4 = j4 + sqrt((p(i,1) - CN(4,1))^2 + (p(i,2) - CN(4,2))^2 + (p(i,3) - CN(4,3))^2);
           end
        end
        JN = j1 + j2 + j3 + j4;                    %四种类别的类内所有点与该类中心的距离和
        e = JN - JO;
         %判断是否接受新解
        if e <= 0
            JO = JN;CO = CN;IDXO = IDXN;
        else
%       if(rand < exp( - e/T))
%           JO = JN;
%           CO = CN;
%           IDXO = IDXN;
%       else
%           IDXN = IDXO;IDX = IDXO;CN = CO;JN = JO;
%       end
%    end
%    else
      IDXN = IDXO;IDX = IDXO;CN = CO;
      JN = JO;
    end
  end
%内层循环结束
     T = T * 0.9;
```

```
%       if(T == 0)
%         break;
%       end
      time = time + 1;
%     if(time - timeb > 1000)
%         break;
%     end
   disp('已退火次数');
   A = time - 1
   disp('最优目标函数值');
   J = JO
end
time1 = toc                  %退火需要的时间
hold on;
plot3(CO(:,1),CO(:,2),CO(:,3),'o');grid;box
%title('蚁群聚类结果(R = 100,t = 10000)')
xlabel('X')
ylabel('Y')
zlabel('Z')
index1 = find(IDXN == 1)
index2 = find(IDXN == 2)
index3 = find(IDXN == 3)
index4 = find(IDXN == 4)
plot3(p(index1,1),p(index1,2),p(index1,3),'r + ');grid;
plot3(p(index2,1),p(index2,2),p(index2,3),'g * ');grid;
plot3(p(index3,1),p(index3,2),p(index3,3),'kx');grid;
plot3(p(index4,1),p(index4,2),p(index4,3),'m.');grid;
```

程序运行完以后,出现如图 7-4 所示数据分类图。

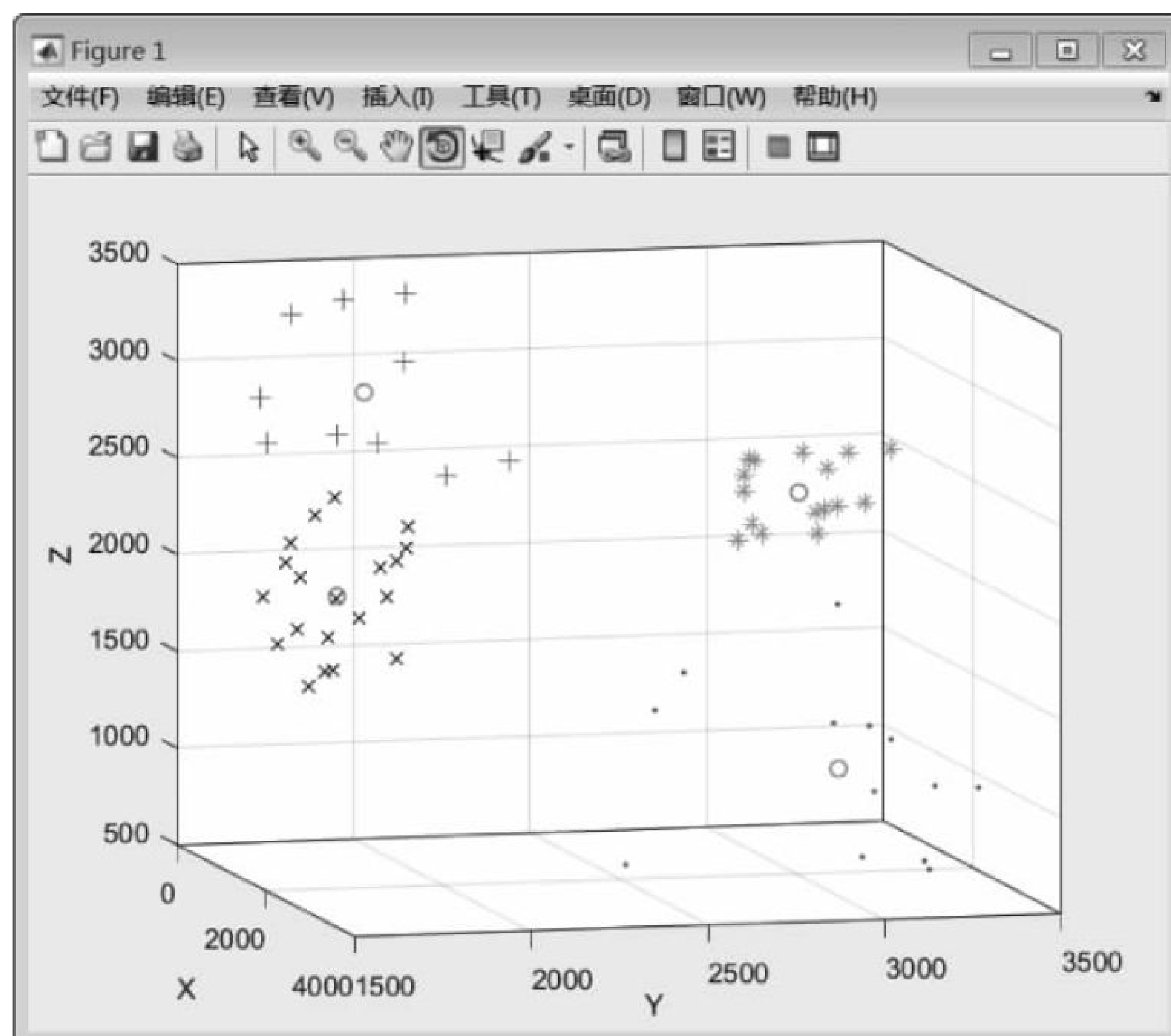


图 7-4 模拟退火分类结果图

MATLAB 程序的运行结果如下：

```
index1 =
4    6    16    25    32    39    42    53    54    56
index2 =
1～15 列
2    5    9    10    12    13    23    27    28    29    34   38   44   46
48
16 列
55
index3 =
1～15 列
1    3    7    11    17    20    21    26    30    31    33   37   40   41
47
16～20 列
51   52   57    58    59
index4 =
8    14   15    18    19    22    24    35    36    43    45   49   50
```

将分类结果与原始分类结果对比，如表 7-2 所示，可以发现分类效果很好。

表 7-2　模拟退火分类与原始分类结果对比

序　号	A	B	C	原始分类结果	模拟退火网络分类结果
1	1739.94	1675.15	2395.96	3	3
2	373.3	3087.05	2429.47	4	4
3	1756.77	1652	1514.98	3	3
4	864.45	1647.31	2665.9	1	1
5	222.85	3059.54	2002.33	4	4
6	877.88	2031.66	3071.18	1	1
7	1803.58	1583.12	2163.05	3	3
8	2352.12	2557.04	1411.53	2	2
9	401.3	3259.94	2150.98	4	4
10	363.34	3477.95	2462.86	4	4
11	1571.17	1731.04	1735.33	3	1
12	104.8	3389.83	2421.83	4	4
13	499.85	3305.75	2196.22	4	4
14	2297.28	3340.14	535.62	2	2
15	2092.62	3177.21	584.32	2	2
16	1418.79	1775.89	2772.9	1	1
17	1845.59	1918.81	2226.49	3	1
18	2205.36	3243.74	1202.69	2	2
19	2949.16	3244.44	662.42	2	2
20	1692.62	1867.5	2108.97	3	1
21	1680.67	1575.78	1725.1	3	3
22	2802.88	3017.11	1984.98	2	2
23	172.78	3084.49	2328.65	4	4

续表

序　号	A	B	C	原始分类结果	模拟退火网络分类结果
24	2063.54	3199.76	1257.21	2	2
25	1449.58	1641.58	3405.12	1	1
26	1651.52	1713.28	1570.38	3	2
27	341.59	3076.62	2438.63	4	4
28	291.02	3095.68	2088.95	4	4
29	237.63	3077.78	2251.96	4	4
30	1702.8	1639.79	2068.74	3	3
31	1877.93	1860.96	1975.3	3	3
32	867.81	2334.68	2535.1	1	1
33	1831.49	1713.11	1604.68	3	3
34	460.69	3274.77	2172.99	4	4
35	2374.98	3346.98	975.31	2	2
36	2271.89	3482.97	946.7	2	2
37	1783.64	1597.99	2261.31	3	3
38	198.83	3250.45	2445.08	4	4
39	1494.63	2072.59	2550.51	1	1
40	1597.03	1921.52	2126.76	3	3
41	1598.93	1921.08	1623.33	3	3
42	1243.13	1814.07	3441.07	1	1
43	2336.31	2640.26	1599.63	2	2
44	354	3300.12	2373.61	4	4
45	2144.47	2501.62	591.51	2	2
46	426.31	3105.29	2057.8	4	4
47	1507.13	1556.89	1954.51	3	3
48	343.07	3271.72	2036.94	4	4
49	2201.94	3196.22	935.53	2	2
50	2232.43	3077.87	1298.87	2	2
51	1580.1	1752.07	2463.04	3	3
52	1962.4	1594.97	1835.95	3	3
53	1495.18	1957.44	3498.02	1	1
54	1125.17	1594.39	2937.73	1	1
55	24.22	3447.31	2145.01	4	4
56	1269.07	1910.72	2701.97	1	1
57	1802.07	1725.81	1966.35	3	3
58	1817.36	1927.4	2328.79	3	3
59	1860.45	1782.88	1875.13	3	3

7.4 结论

虽然模拟退火算法有限度地接受劣解，可以跳出局部最优解，但它明显存在两个弱点：

(1) 如果降温过程足够慢，所得到解的性能会较好，但算法收敛速度太慢。

（2）如果降温过程过快，很可能得不到全局最优解。为此，模拟退火算法的改进及其在各类复杂系统建模及优化问题中的应用仍有大量的内容值得研究。

习题

（1）简述模拟退火过程。

（2）简述模拟退火算法的基本原理。

（3）简述模拟退火算法的组成。

第8章

遗传算法聚类设计

遗传算法是一种模拟自然进化的优化搜索算法。由于它仅依靠适应度函数就可以搜索最优解，不需要有关问题解空间的知识，并且适应度函数不受连续可微等条件的约束，因此在解决多维、高度非线性的复杂优化问题中得到了广泛应用和深入研究。

遗传算法在模式识别、神经网络、机器学习、工业优化控制、自适应控制、生物科学、社会科学等方面都得到了应用。

本章给出了一种基于遗传算法的聚类分析方法。采用浮点数编码方式对聚类的中心进行编码，并用特征向量与相应聚类中心的欧氏距离的和来判断聚类划分的质量，通过选择、交叉和变异操作对聚类中心的编码进行优化，得到使聚类划分效果最好的聚类中心。实验结果显示，该方法的聚类划分能得到比较满意的效果。由于前面章节已经对聚类算法做了详细介绍，此处不再赘述。

8.1 遗传算法简介

遗传算法的研究历史可以追溯到20世纪60年代，从20世纪60年代到70年代中期，是遗传算法的萌芽期。遗传算法的基本原理最早由美国科学家J. H. Holland在1962年提出；1967年，J. D. Bagay在他的博士论文中首次使用了遗传算法这个术语；1975年，J. H. Holland在他出版的专著《自然界和人工系统的适应性》中详细地介绍了该算法，为其奠定了数学基础，人们常常把这一事件视为遗传算法正式得到承认的标志。此标志说明遗传算法已经完成孕育过程，Holland也被视为该算法的创始人。

从20世纪70年代中期到80年代，遗传算法得到了不断完善，属于遗传算法的成长期。这一时期相继出现了有关遗传算法的博士论文，分别研究了遗传算法在函数优化，组合优化中的应用，并从数学上探讨了遗传算法的收敛性，对遗传算法的发展起到了很大的推动作用。30多年来，该算法不论是实际应用还是建模，其范围不断扩大，而算法本身也渐渐成熟，形成了算法的大体框架。其后出现的遗传算法的许多改进研究，大体都遵循了这个框架。

20 世纪 80 年代末以来是遗传算法的蓬勃发展期，这不仅表现在理论研究方面，还表现在应用领域方面。随着遗传算法研究和应用的不断深入，一系列以遗传算法为主题的国际会议十分活跃：开始于 1985 年的国际遗传算法会议（International Conference on Genetic Algorithm，ICGA）每两年举办一次。在欧洲，从 1990 年开始也每隔一年举办一次类似的会议。这些会议的举办表明遗传算法正不断地引起学术界的重视，同时这些会议的论文集中反映了遗传算法近年来的最新发展和动向。

随着计算速度的提高和并行计算的发展，遗传算法的速度已经不再是制约其应用的因素，遗传算法已在机器学习、过程控制、图像处理、经济管理等领域取得了巨大成功，但如何将各专业知识融入遗传算法中，目前仍在继续研究。

8.2 遗传算法原理

遗传算法是一种搜索最优解方法，其过程类似于自然进化，通过作用于染色体上的基因寻找好的染色体来求解问题。遗传算法模拟生物进化的过程，具有很好的自组织、自适应和自学习能力，在求解大规模优化问题的全局最优解方面具有广泛的应用。聚类问题的解是各个集合的中心，通过对问题的解进行编码，然后对编码进行选择、交叉和变异操作，结合评价解好坏的适应度函数，就可找到按所选的评价标准来说是比较好的聚类划分。

遗传算法对于复杂的优化问题无须建模和复杂运算，只要利用遗传算法的三种算子就能得到最优解，这就是遗传算法的基本原理。

8.2.1 遗传算法的基本术语

由于遗传算法是自然遗传学和计算机科学相互结合渗透而成的新的计算方法，因此遗传算法中经常使用自然进化中有关的一些基本术语。了解这些用语对理解遗传算法是十分必要的。

（1）染色体（chromosome），又称为个体（individual）。生物的染色体是由基因（gene）构成的位串，包含了生物的遗传信息。遗传算法中的染色体对应的是数据或数组，通常是由一维的串结构数据来表示的。串结构上每个位置上的数据对应一个基因，而各位置上所取的值对应于基因值。

（2）编码（coding）。把问题的解表示为位串的过程称为编码，编码后的每个位串就表示一个个体，即问题的一个解。

（3）种群（population）。由一定数量的个体组成的群体，也就是问题的一些解的集合。种群中个体的数量称为种群规模。

（4）适应度（fitness）。评价群体中个体对环境适应能力的指标，就是解的好坏，由评价函数 F 计算得到。在遗传算法中，F 是求解问题的目标函数，也就是适应度函数。

（5）遗传算子（genetic operator）。产生新个体的操作，常用的遗传算子有选择、交叉和变异等。

选择（selection）：以一定概率从种群中选择若干个体的操作。一般而言，该操作是基于

适应度进行的,适应度越高的个体,产生后代的概率就越高。

交叉(crossover):把两个串的部分基因进行交换,产生两个新串作为下一代的个体。交叉概率(P_c)决定两个个体交叉操作的可能性。

变异(mutation):随机地改变染色体的部分基因,例如把 0 变 1,或把 1 变 0,产生新的染色体。

8.2.2 遗传算法进行问题求解的过程

遗传算法进行问题求解过程如下。

(1) 选择编码策略,参数编码,把参数集合和域转换为位串结构空间。

(2) 适应度函数的设计。

(3) 确定遗传策略,包括群体规模,选择、交叉、变异算子及其概率。

(4) 初始群体的设定。

(5) 计算群体中各个体的适应度值。

(6) 按照遗传策略,将遗传算子作用于种群,产生下一代种群。

(7) 迭代终止判定。

遗传算法涉及六大要素:参数编码、初始群体的设定、适应度函数的设计、遗传操作的设计、控制参数的设定、迭代终止条件。

8.2.3 遗传算法的优缺点

遗传算法的优点如下。

(1) 快速简单。搜索过程具有快速随机的搜索能力。

(2) 并行性。搜索具有潜在的并行性,可以进行多个个体的同时比较,健壮性好。

(3) 随机性。使用概率机制进行迭代。

(4) 可扩展性。容易与其他算法结合。

遗传算法的缺点如下。

(1) 早熟。这是最大的缺点,即算法对新空间的探索能力是有限的,也容易收敛到局部最优解。

(2) 大量计算。涉及大量个体的计算,当问题复杂时,计算时间是个问题。

(3) 处理规模小。目前对于维数较高的问题,还是很难处理和优化。

(4) 难于处理非线性约束。对非线性约束的处理,大部分算法都是添加惩罚因子,这是一笔不小的开支。

(5) 稳定性差。因为算法属于随机类算法,需要多次运算,结果的可靠性差,不能稳定地得到解。

8.2.4 遗传算法的基本要素

遗传算法包含如下 5 个基本要素:问题编码、初始群体的设定、适应度函数的设计、遗

传操作设计和控制参数的设定。这5个要素构成了遗传算法的核心内容。

1. 问题编码

编码机制是遗传算法的基础。通常遗传算法不直接处理问题空间的数据，而是将各种实际问题变换为与问题无关的串个体。不同串长和不同的编码方式，对问题求解的精度和遗传算法的求解效率有着很大的影响，因此针对一个具体应用问题，应考虑多方面因素，以寻求一种描述方便、运行效率高的编码方案。迄今为止，遗传算法常采用的编码方法主要有两类：二进制编码和浮点数编码。

1）二进制编码

二进制编码是遗传算法中最常用的一种编码方法，该方法使用的编码符号集是由二进制符号0和1所组成的二值符号集{0,1}，它所构成的个体是一个二进制编码符号串。二进制编码符号串的长度与问题所要求的求解精度有关。该编码方法具有操作简单、易于实现等特点。

2）浮点数编码

浮点数编码方法又叫真值编码方法，它是指个体的每个基因值用某一范围内的一个浮点数来表示，个体的编码长度等于其决策变量的个数。该编码方法具有适用于大空间搜索、局部搜索能力强、不易陷入局部极值、收敛速度快的特点。

2. 初始群体的生成

遗传算法处理流程中，编码设计之后的任务是初始群体的设定，并以此为起点进行一代一代的进化直到按照某种进化终止准则终止。最常用的初始方法是无指导的随机初始化。

3. 适应度函数(fitness function)的确定

在遗传算法中，按与个体适应度成正比的概率来决定当前群体中的每个个体遗传到下一代群体中的机会多少，一般希望适应值越大越好，且要求适应值非负。因此适应值函数的选取至关重要，它直接影响到算法的收敛速度及最终能否找到最优解。

适应度函数是根据目标函数确定的，针对不同种类的问题，目标函数有正有负，因此必须确定由目标函数值到适应度函数之间的映射规则，以适应上述的要求。适应度函数的设计应满足以下条件：

(1) 单值、连续、非负、最大化。

(2) 计算量小。适应度函数设计尽可能简单，以减少计算的复杂性。

(3) 通用性强。适应度对某类问题应尽可能通用。

4. 遗传操作

遗传操作主要包括选择、交叉、变异三个算子。关于每个算子的作用前面已提及了，此处不再赘述。这里主要说一下每种算子常用的方法。

1）选择算子

在适应度计算之后是实际的选择，选择的目的是为了从当前群体中选出优良的个体，使

它们作为父代进行下一代繁殖。采用基于适应度的选择原则,适应度越强被选中概率越大,体现了优胜劣汰的进化机制。这里介绍几种常用的选择方法:

(1) 赌轮选择法。该方法中个体被选中的概率与其适应度大小成正比。

(2) 最优保存策略。群体中适应度最高的个体不进行交叉变异,用它替换下一代种群中适应度最低的个体。

(3) 锦标赛选择法。随机从种群中选取一定数目的个体,然后将适应度最高的个体遗传到下一代群体中。这个过程重复进行直到完成个体的选择。

(4) 排序选择法。根据适应度对群体中的个体排序,然后把事先设定的概率表分配给个体,作为各自的选择概率。这样,选择概率和适应度无关而仅与序号有关。

选择算子确定的好坏,直接影响到遗传算法的计算结果。如果选择算子确定不当,会导致进化停滞不前或出现早熟问题。选择策略与编码方式无关。

2) 交叉算子

交叉算子是遗传算法中最主要的遗传操作,也是遗传算法区别于其他进化运算的重要特征,通过交叉操作可以产生新的个体。该操作模拟了自然界生物体的突变,体现了信息交换思想,决定着遗传算法的收敛性和全局搜索能力。

交叉算子的设计与实现与所研究的问题密切相关,一般要求它既不要破坏原个体的优良性,又能够产生出一些较好的新个体,而且,还要和编码设计一同考虑。目前适合于二进制编码的个体和浮点数编码的个体的交叉算法主要有:

(1) 单点交叉。单点交叉又称简单交叉,是指在个体编码串中随机设置一个交叉点,实行交叉时,在该点相互交换两个配对个体的部分染色体。

(2) 两点交叉与多点交叉。两点交叉是指在个体编码串中随机设置了两个交叉点,交换两个个体在所设定两个交叉点之间的部分染色体。例如:

$$A: 10 \mid 110 \mid 11 \Rightarrow A' = 1001011$$

$$B: 00 \mid 010 \mid 00 \Rightarrow B' = 0011000$$

多点交叉是两点交叉的推广。

(3) 均匀交叉。均匀交叉也称一致交叉,是指两个交叉个体的每个基因都以相同的交叉概率进行交换,从而形成两个新的个体。

(4) 算术交叉。该算法是指由两个个体的线性组合而产生的两个新的个体。该方法的操作对象一般是由浮点数编码产生的个体。

3) 变异算子

选择和交叉算子基本上完成了遗传算法的大部分搜索功能,变异操作只是对产生的新个体起辅助作用,但是它必不可少,因为变异操作决定了遗传算法的局部搜索能力。变异算子与交叉算子相互配合,共同完成对搜索空间的全局搜索和局部搜索,从而使得遗传算法能够以良好的搜索性能找到最优解。

目前适合于二进制编码的个体和浮点数编码的个体的变异算法主要有:

(1) 基本位变异。该算法是指对群体中的个体编码串根据变异概率,随机挑选一个或多个基因位并对这些基因位的基因值进行变动。例如:

个体 A:1011011

指定第三位为变异位，则

个体 A′：1001011

(2) 均匀变异。该算法是指分别用符合某一范围内均匀分布的随机数，以某一较小的概率来替换个体编码串中各基因座上原有的基因值。

(3) 边界变异。该算法是均匀变异的一个变形。在进行边界变异时，随机选取基因座的两个对应边界基因值之一去替换原有的基因值。

(4) 高斯近似变异。该算法是指进行变异操作时用符合均值为 P，方差为 P^2 的正态分布的一个随机数来替换原有的基因值。

5. 控制参数

控制参数主要有群体规模、迭代次数、交叉概率、变异概率等，对于它们基本的遗传算法都需要提前设定。

N：群体大小，即群体中所含个体的数量。如果群体规模大，可提供大量模式，使遗传算法进行启发式搜索，防止早熟发生，但会降低效率；如果群体规模小，可提高速度，但却会降低效率。一般取值为 20～100。

T：遗传运算的终止进化代数，一般取值为 100～500。

P_c：交叉概率。它影响着交叉算子的使用频率，交叉率越高，可以越快地收敛到全局最优解，因此一般选择较大的交叉率。但如果交叉率太高，也可能导致过早收敛，而交叉率太低，可能导致搜索停滞不前，一般取值为 0.4～0.99。

P_m：变异概率。变异率控制着变异算子的使用频率，它的大小将影响群体的多样性及成熟前的收敛性能。变异率的选取一般受种群大小、染色体长度等因素影响，通常选取很小的值。但变异率太低可能使某基因值过早丢失、信息无法恢复；变异率太高，遗传算法可能会变成随机搜索。一般取值为 0.0001～0.1。

这四个运行参数对遗传算法的求解结果和求解效率都有一定的影响，但目前尚无合理选择它们的理论依据。在实际应用中，常常需要经过多次实验后才能确定参数或其范围。

8.3 算法实现

本例使用表 1-1 中的三元色数据，希望按照颜色数据所表征的特点，将数据按照各自所属的类别进行归类。

下面取表 1-1 的 59 组数据为分析对象，使用 MATLAB 构建遗传优化算法。程序流程如图 8-1 所示。

8.3.1 种群初始化

遗传聚类算法需要设置的参数有四个，分别是：交叉概率 pcross、遗传概率 pmutation、

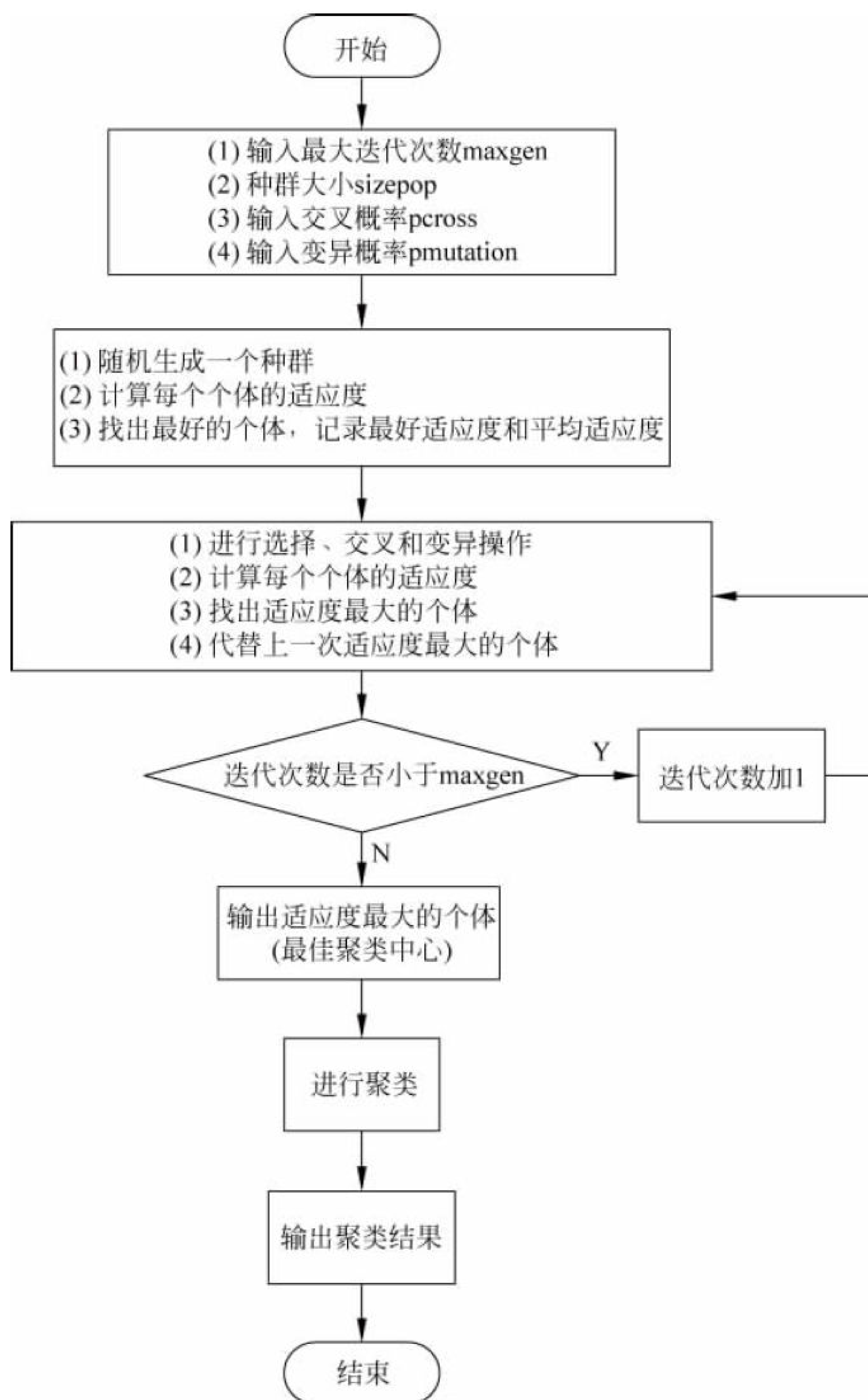


图 8-1 遗传聚类算法流程图

进化代数(迭代次数)maxgen 和种群规模 sizepop。这里的参数的设置如 MATLAB 程序代码所示：

```
%% 参数初始化
maxgen = 100;           % 进化代数,即迭代次数,初始预定值选为 100
sizepop = 100;          % 种群规模,初始预定值选为 100
pcross = 0.9;           % 交叉概率选择,0～1,一般取 0.9
pmutation = 0.01;       % 变异概率选择,0～1,一般取 0.01
```

按照遗传算法的程序流程,用遗传算法求解,首先要解决的问题是如何确定编码和解码运算。编码形式决定了交叉算子和变异算子的操作方式,并对遗传算法的性能如搜索能力和计算效率等影响很大。

由8.2节可知，遗传算法常用的编码方法有浮点数编码和二进制编码两种。由于聚类样本具有多维性、数据量大的特点，如果采用传统的二进制编码，染色体的长度会随着维数的增加或精度的提高而显著增加，从而使搜索空间急剧增大，大大降低了计算效率。基于上面的分析，这里采用浮点数编码方法。

在遗传聚类问题中，可采用的染色体编码方式有两种：一种是按照数据所属的聚类划分来生成染色体的整数编码方式；另一种是把聚类中心（聚类原型矩阵）作为染色体的浮点数编码。由于聚类问题的解是各聚类中心，因此本文采用基于聚类中心的浮点数编码。

所谓的将聚类中心作为染色体的浮点数编码，就是把一条染色体看成是由 K 个聚类中心组成的一个串。具体编码方式如下：对于 D 维样本数据的 K 类聚类分析，基于聚类中心的染色体结构为

$$S=\{x_{11},x_{12},\cdots,x_{1d},x_{21},x_{22},\cdots,x_{2d},\cdots,x_{k1},x_{k2},\cdots,x_{kd}\} \tag{8-1}$$

即每条染色体都是一个长度为 $k\times d$ 的浮点码串。这种编码方式意义明确、直观，避免了二进制编码在运算过程中反复进行译码、解码以及染色体长度受限等问题。

确定了编码方式之后，接下来要进行种群初始化。初始化的过程是随机产生一个初始种群的过程。首先从样本空间中随机选出 K 个个体，K 值由用户自己来指定，每个个体表示一个初始聚类中心，然后根据我们所采用的编码方式将这组个体（聚类中心）编码成一条染色体。然后重复进行 P_{size} 次染色体初始化（P_{size} 为种群大小），直到生成初始种群。

8.3.2 适应度函数的设计

根据前面的介绍可知，遗传算法中的适应度函数是用来评价个体的适应度、区别群体中个体优劣的标准。个体的适应度越高，其存活的概率就越大。聚类问题实际上就是找到一种划分，该划分使待聚类数据集的目标函数 G_c（$G_c=\sum_{j=1}^{c}\sum_{k=1}^{n_j}\|x_k^{(j)}-m_j\|^2$，$m_j(j=1,2,\cdots,c)$ 是聚类中心，x_k 是样本）达到最小。遗传算法在处理过程中根据每个染色体（K 个聚类中心）进行聚类划分，根据每个聚类中的点与相应聚类中心的距离作为判别聚类划分质量好坏的准则函数 G_c，G_c 越小表示聚类划分的质量越好。

遗传算法的目的是搜索使目标函数 G_c 值最小的聚类中心，因此可借助目标函数来构造适应度函数

$$\text{fit}=\frac{1}{G_c} \tag{8-2}$$

由式(8-2)可以看出，目标函数值越小的聚类中心，其适应度也就越大；目标函数值越大的聚类中心，其适应度也就越低。

种群初始化的MATLAB程序代码如下：

```
individuals = struct('fitness',zeros(1,sizepop),'chrom',[]);
% 种群,种群由 sizepop 条染色体(chrom)及每条染色体的适应度(fitness)组成
avgfitness = [];
% 记录每一代种群的平均适应度,首先赋给一个空数组
bestfitness = [];
```

```
%记录每一代种群的最佳适应度,首先赋给一个空数组
bestchrom = [];
%记录适应度最好的染色体,首先赋给一个空数组

%初始化种群
for i = 1:sizepop
    %随机产生一个种群
individuals.chrom(i,:) = 4000 * rand(1,12);
%把 12 个 0～4000 的随机数赋给种群中的一条染色体,代表 K = 4 个聚类中心
   x = individuals.chrom(i,:);
    %计算每条染色体的适应度
   individuals.fitness(i) = fitness(x);
end

%%找最好的染色体
[bestfitness bestindex] = max(individuals.fitness);
%找出适应度最大的染色体,并记录其适应度的值(bestfitness)和染色体所在的位置(bestindex)
bestchrom = individuals.chrom(bestindex,:);
%把最好的染色体赋给变量 bestchrom
avgfitness = sum(individuals.fitness)/sizepop;
%计算群体中染色体的平均适应度

% 记录每一代进化中最好的适应度和平均适应度
trace = [avgfitness bestfitness];
```

适应度函数的 MATLAB 程序代码如下：

```
function fit = fitness(x)
%% 计算个体适应度值
%x input 个体
%fit output 适应度值
data = load["a.txt"];
kernel = [x(1:3);x(4:6);x(7:9);x(10:12)];
%对染色体进行编码,其中 x(1:3)代表第一个聚类中心,x(4:6)代表第二个聚类中心,x(7:9)代表第
%三个聚类中心,x(10:12)代表第四个聚类中心
Gc = 0;
%Gc 代表聚类的准则函数
[n,m] = size(data);
%求出待聚类数据的行和列
for i = 1:n
      dist1 = norm(data(i,1:3) - kernel(1,:));
      dist2 = norm(data(i,1:3) - kernel(2,:));
      dist3 = norm(data(i,1:3) - kernel(3,:));
      dist4 = norm(data(i,1:3) - kernel(4,:));
      %计算待聚类数据中的某一点到各个聚类中心的距离
     a = [dist1 dist2 dist3 dist4];
mindist = min(a);
%取其中的最小值,代表其被划分到某一类
Gc = mindist + Gc;
```

```
%求类中某一点到其聚类中心的距离和,即准则函数
end
fit = 1/Gc;
%求出染色体的适应度,即准则函数的倒数,聚类的准则函数越小,染色体的适应度越大,聚类的效
%果也就越好
```

8.3.3 选择操作

在生物进化的过程中,对生存环境适应能力强的物种将有更多的机会遗传到下一代,而适应能力差的物种遗传到下一代的机会就相对较少。遗传算法中的选择操作体现了这一“适者生存”的原则:适应度越高的个体,参与后代繁殖的概率越高。遗传算法中的选择操作就是用来确定如何从父代群体中按照某种方法选取哪些个体遗传到下一代群体的一种遗传运算。选择操作建立在对个体的适应度进行评价的基础之上。进行选择操作的目的是为了避免基因缺失、提高全局收敛性和计算效率。

为了保证适应度最好的染色体保留到下一代群体而不被遗传操作破坏掉,根据遗传算法中目前已有的选择方法,本例采用了轮盘赌选择算子。该选择算子具体选择步骤如下:

(1) 首先在计算完当前种群的适应度后,记录下其中适应度最大的个体。

(2) 根据各个体的适应度值 $f(S_i)(i=1,2,\cdots,P_{\text{size}})$ 计算各个体的选择概率

$$P_i = \frac{f(S_i)}{\sum_{i=1}^{P_{\text{size}}} f(S_i)} \tag{8-3}$$

式中,P_{size} 为种群大小,$\sum_{i=1}^{P_{\text{size}}} f(S_i)$ 为所有个体适应度的总和。

(3) 根据计算出的选择概率,使用轮盘赌法选出个体。

(4) 被选出的个体参加交叉、变异操作产生新的群体。

(5) 计算出新群体中的各条染色体的适应度值,用上一代中记录的最优个体替换掉新种群中的最差个体,这样就产生了下一代群体。

这种遗传操作既不断提高了群体的平均适应度值,又保证了最优个体不被破坏,使得迭代过程向最优方向发展。

选择操作的 MATLAB 程序代码如下:

```
function ret = Select(individuals,sizepop)
% 本函数对每一代种群中的染色体进行选择,以进行后面的交叉和变异步骤
% individuals  input:  种群信息
% sizepop      input:  种群规模
% ret          output: 经过选择后的种群

sumfitness = sum(individuals.fitness);
%计算群体的总适应度
sumf = (individuals.fitness)./sumfitness;
%计算出染色体的选择概率,即染色体的适应度除以总适应度
```

```
index = [];
%用来记录被选中染色体的序号,首先付给一个空数组
for i = 1:sizepop
%转 sizepop 次轮盘
pick = rand;
%把一个 0～1 的随机数赋给 pick
    while pick == 0
         pick = rand;
end
%确保 pick 被赋值
    for i = 1:sizepop
        pick = pick - sumf(i);
%染色体的选择概率越大,pick 越容易小于 0,即染色体越容易被选中
        if pick < 0
            index = [index i];
            %把被选中的染色体的序号赋给 index
            break;
        end
    end
end
individuals.chrom = individuals.chrom(index,:);
%记录选中的染色体
individuals.fitness = individuals.fitness(index);
%记录选中染色体的适应度
ret = individuals;
%输出经过选择后的染色体
```

8.3.4 交叉操作

交叉操作是把两个父个体的部分结构加以替换重组而产生新个体的操作,也称为基因重组。交叉的目的是为了能够在下一代产生新的个体,因此交叉操作是遗传算法的关键部分,交叉算子的好坏,在很大程度上决定了算法性能的好坏。

由于染色体以聚类中心矩阵为基因,造成了基因串的无序性。两条染色体的等位基因之间的信息不一定相关,如果采用传统的交叉算子进行交叉,将使染色体在进行交叉时,不能很好地将基因配对起来,使得生成的下一代个体的适应值普遍较差,影响了算法的效率。为了改善这种情况,又因为本例所使用的是浮点数编码方式,因此本例采用了一种以随机交叉为基础的随机交叉算子。

交叉操作的 MATLAB 程序代码如下：

```
function ret = Cross(pcross,chrom,sizepop)
%本函数完成交叉操作
% pcorss     input:  交叉概率
% lenchrom   input:  染色体的长度
% chrom      input:  染色体群
% sizepop    input:  种群规模
% ret        output: 交叉后的染色体
```

```
for i = 1:sizepop
    % 交叉概率决定是否进行交叉
    pick = rand;
    while pick == 0
        pick = rand;
end
% 给 pick 赋予一个 0～1 的随机数
    if pick > pcross
        continue;
    end
    % 当 pick < pcross 时,进行交叉操作
    index = ceil(rand(1,2). * sizepop);
    while (index(1) == index(2)) | index(1) * index(2) == 0
        index = ceil(rand(1,2). * sizepop);
    end
    % 在种群中,随机选择两个个体
    pos = ceil(rand * 3);
    while pos == 0
        pos = ceil(rand * 3);
    end
    %  在染色体当中,随机选择交叉位置
    temp = chrom(index(1),pos);
    chrom(index(1),pos) = chrom(index(2),pos);
    chrom(index(2),pos) = temp;
    % 把两条染色体某个位置的信息进行交叉互换
end
ret = chrom;
% 输出经过交叉操作后的染色体
```

8.3.5 变异操作

在生物自然进化的过程中,细胞分裂的过程可能会出现某些差错,导致基因变异情况的发生。变异操作就是模仿这种情况产生的。所谓变异操作,是指将个体染色体编码串中的某些基因座上的基因值用该基因座的其他等位来替换,从而形成一个新的个体。变异的目的有二：一是增强算法的局部搜索能力；二是增加种群的多样性,改善算法的性能,避免早熟收敛。变异操作既可以产生种群中没有的新基因又可以恢复迭代过程中被破坏的基因。本例所使用的是浮点数编码方式,采用随机变异算子来完成变异操作。

变异操作的 MATLAB 程序代码如下：

```
function ret = Mutation(pmutation,chrom,sizepop)
% 本函数完成变异操作
% pcorss          input:  变异概率
% lenchrom        input:  染色体长度
% chrom           input:  染色体群
% sizepop         input:  种群规模
% bound           input:  每个个体的上届和下届
% ret             output: 变异后的染色体
```

```
for i = 1:sizepop
    % 变异概率决定该轮循环是否进行变异
    pick = rand;
    if pick > pmutation
        continue;
    end
    %当 pick 小于变异概率时,执行变异操作
    pick = rand;
    while pick == 0
        pick = rand;
    end
    index = ceil(pick * sizepop);
    %在种群中,随机选择一条染色体
    pick = rand;
    while pick == 0
        pick = rand;
    end
    pos = ceil(pick * 3);
    %在染色体当中,随机选择变异位置
chrom(index, pos) = rand * 4000;
%染色体进行变异
end
ret = chrom;
%输出变异后的染色体
```

8.3.6 完整程序及仿真结果

遗传算法的完整 MATLAB 程序代码如下：

```
clc
tic
%%参数初始化
maxgen = 100;       %进化代数,即迭代次数,初始预定值选为 100
sizepop = 100;      %种群规模,初始预定值选为 100
pcross = 0.9;       %交叉概率选择,0~1,一般取 0.9
pmutation = 0.01;   %变异概率选择,0~1,一般取 0.01
individuals = struct('fitness',zeros(1,sizepop),'chrom',[]);
%种群,种群由 sizepop 条染色体(chrom)及每条染色体的适应度(fitness)组成
avgfitness = [];
%记录每一代种群的平均适应度,首先赋予一个空数组
bestfitness = [];
%记录每一代种群的最佳适应度,首先赋予一个空数组
bestchrom = [];
%记录适应度最好的染色体,首先赋予一个空数组
%初始化种群
for i = 1:sizepop
%随机产生一个种群
```

```
individuals.chrom(i,:) = 4000 * rand(1,12);
%把 12 个 0～4000 的随机数赋予种群中的一条染色体,代表 K = 4 个聚类中心
x = individuals.chrom(i,:);
%计算每条染色体的适应度
individuals.fitness(i) = fitness(x);
end
%%找最好的染色体
[bestfitness bestindex] = max(individuals.fitness);
%找出适应度最大的染色体,并记录其适应度的值(bestfitness)和染色体所在的位置(bestindex)
bestchrom = individuals.chrom(bestindex,:);
%把最好的染色体赋予变量 bestchrom
avgfitness = sum(individuals.fitness)/sizepop;
%计算群体中染色体的平均适应度
trace = [avgfitness bestfitness];
%记录每一代进化中最好的适应度和平均适应度
for i = 1:maxgen
i
%输出进化代数
individuals = Select(individuals,sizepop);
avgfitness = sum(individuals.fitness)/sizepop;
%对种群进行选择操作,并计算出种群的平均适应度
individuals.chrom = Cross(pcross,individuals.chrom,sizepop);
%对种群中的染色体进行交叉操作
individuals.chrom = Mutation(pmutation,individuals.chrom,sizepop);
%对种群中的染色体进行变异操作
for j = 1:sizepop
x = individuals.chrom(j,:); %解码
[individuals.fitness(j)] = fitness(x);
end
%计算进化种群中每条染色体的适应度
[newbestfitness,newbestindex] = max(individuals.fitness);
[worestfitness,worestindex] = min(individuals.fitness);
%找到最小和最大适应度的染色体及它们在种群中的位置
if bestfitness < newbestfitness
bestfitness = newbestfitness;
bestchrom = individuals.chrom(newbestindex,:);
end
%代替上一次进化中最好的染色体
individuals.chrom(worestindex,:) = bestchrom;
individuals.fitness(worestindex) = bestfitness;
%淘汰适应度最差的个体
avgfitness = sum(individuals.fitness)/sizepop;
trace = [trace;avgfitness bestfitness];
%记录每一代进化中最好的适应度和平均适应度
end
figure(1)
plot(trace(:,1),'-*r');
title('适应度函数曲线(100*100)')
hold on
plot(trace(:,2),'-ob');
```

```
legend('平均适应度曲线','最佳适应度曲线','location','southeast')
%% 画出适应度变化曲线
clc
%% 画出聚类点
data1 = load('aa.txt');
% 待分类的数据
kernel = [bestchrom(1:3);bestchrom(4:6);bestchrom(7:9);bestchrom(10:12)];
% 解码出最佳聚类中心
[n,m] = size(data1);
% 求出待聚类数据的行数和列数
index = cell(4,1);
% 用来保存聚类类别
dist = 0;
% 用来计算准则函数
for i = 1:n
dis(1) = norm(kernel(1,:) - data1(i,:));
dis(2) = norm(kernel(2,:) - data1(i,:));
dis(3) = norm(kernel(3,:) - data1(i,:));
dis(4) = norm(kernel(4,:) - data1(i,:));
% 计算出待聚类数据中的一点到各个聚类中心的距离
[value,index1] = min(dis);
% 找出最短距离和其聚类中心的种类
cid(i) = index1;
% 用来记录数据被划分到的类别
index{index1,1} = [index{index1,1} i];
dist = dist + value;
% 计算准则函数
end
cid;
dist;
%% 绘图
figure(2)
plot3(bestchrom(1),bestchrom(2),bestchrom(3),'ro');
title('result100 * 100')
hold on
% 画出第一类的聚类中心
index1 = index{1,1};
for i = 1:length(index1)
plot3(data1(index1(i),1),data1(index1(i),2),data1(index1(i),3),'r * ')
hold on
end
hold on
% 画出被划分到第一类中的各点
index1 = index{2,1};
plot3(bestchrom(4),bestchrom(5),bestchrom(6),'bo');
hold on
% 画出第二类的聚类中心
for i = 1:length(index1)
plot3(data1(index1(i),1),data1(index1(i),2),data1(index1(i),3),'b * ');
grid on;
```

```
hold on
end
% 画出被划分到第二类中的各点
index1 = index{3,1};
plot3(bestchrom(7),bestchrom(8),bestchrom(9),'go');
hold on
% 画出第三类的聚类中心
for i = 1:length(index1)
plot3(data1(index1(i),1),data1(index1(i),2),data1(index1(i),3),'g * ');
hold on
end
% 画出被划分到第三类中的各点
index1 = index{4,1};
plot3(bestchrom(10),bestchrom(11),bestchrom(12),'ko');
hold on
% 画出第四类的聚类中心
for i = 1:length(index1)
plot3(data1(index1(i),1),data1(index1(i),2),data1(index1(i),3),'k * ');
hold on
end
% 画出被划分到第四类中的各点
toc
```

程序运行完以后，初始聚类结果如图 8-2 所示，适应度曲线如图 8-3 所示。

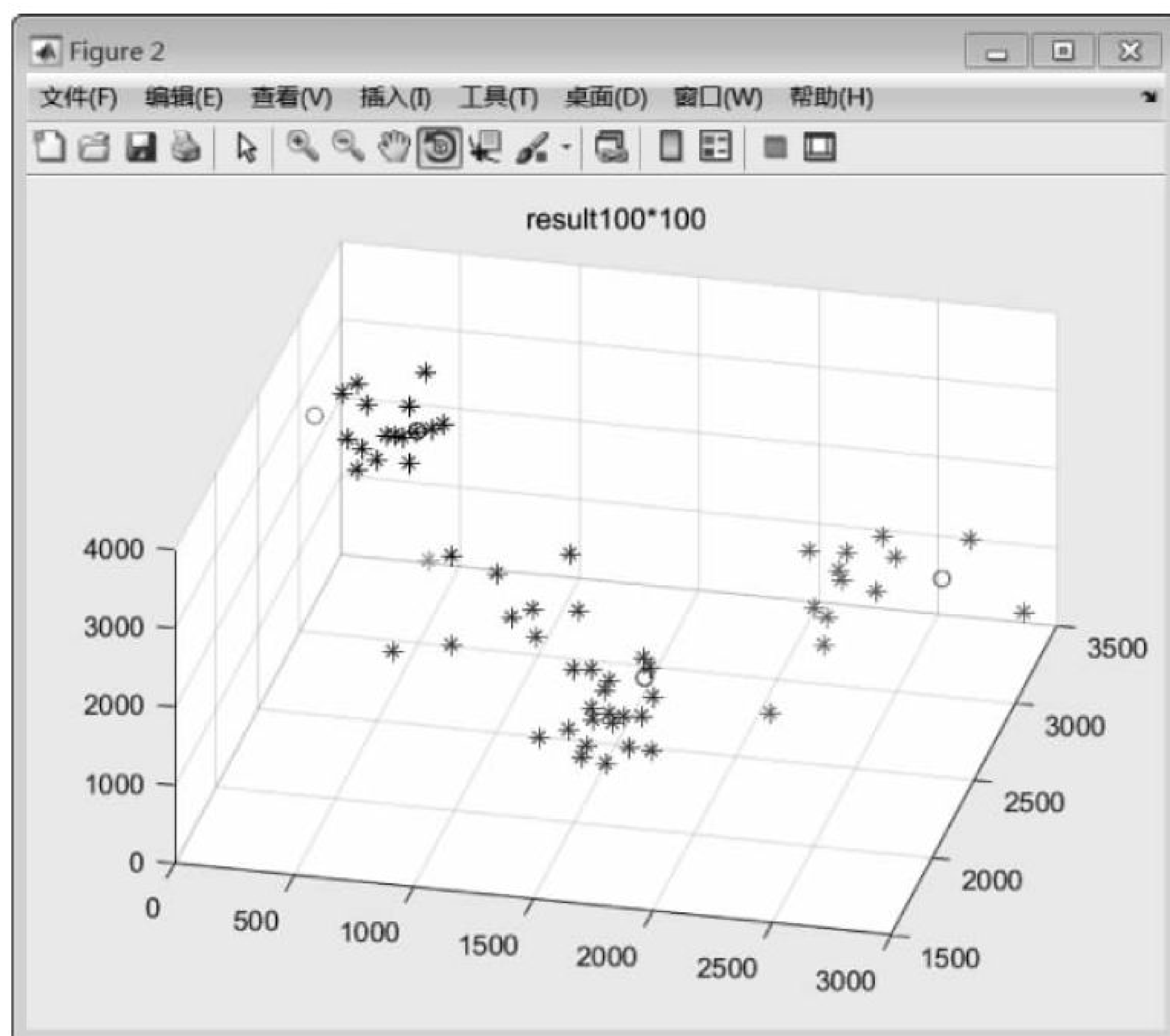


图 8-2 maxgen＝100，sizepop＝100 时的聚类结果

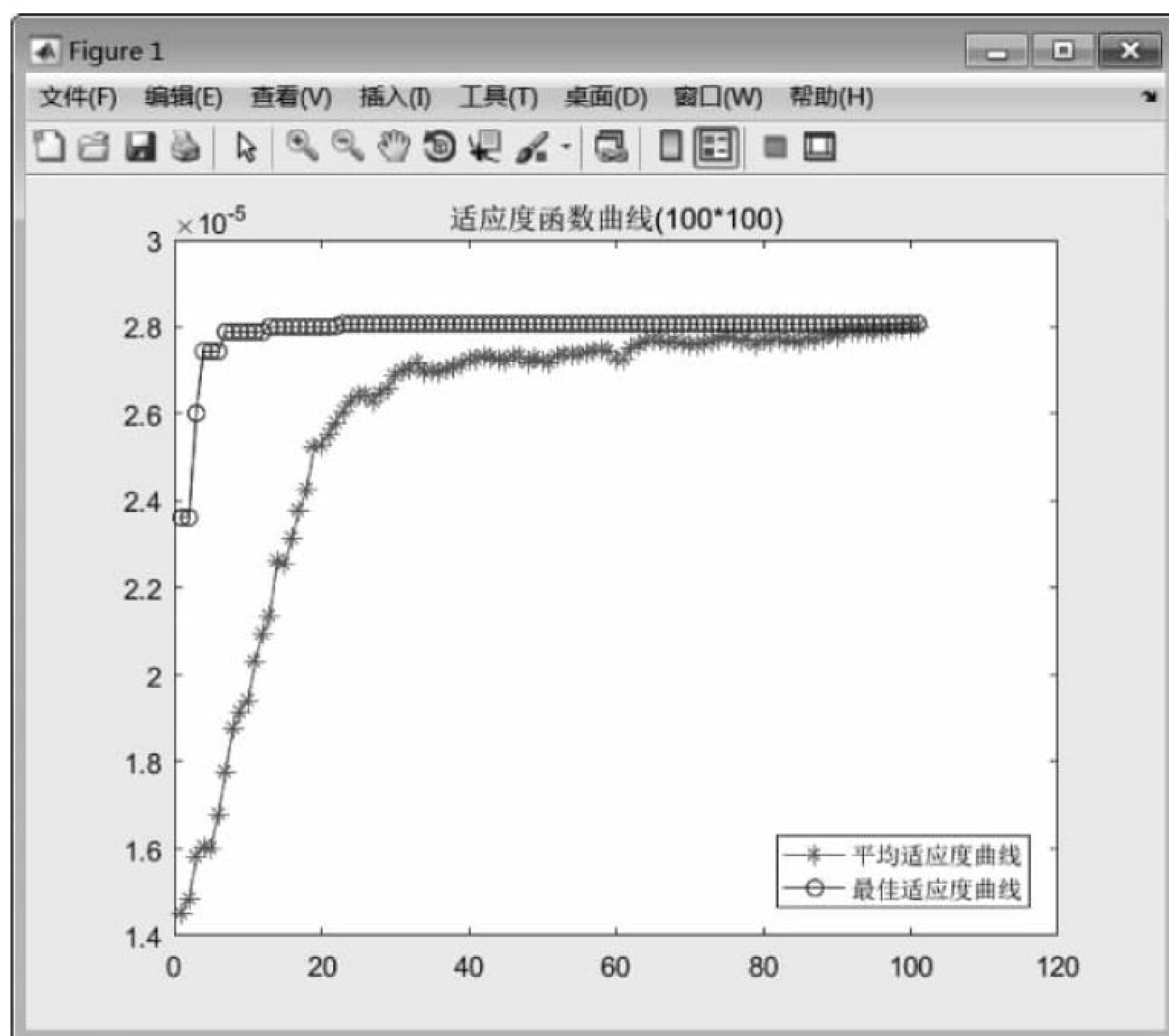


图 8-3 maxgen=100,sizepop=100 时的适应度函数曲线

分析聚类结果可知,当进化代数(即迭代次数)maxgen=100,种群规模 sizepop=100 时,有一类仅有一个数据,聚类结果明显是错误的,并没有按照要求把数据聚为四类。通过分析适应度函数曲线可知,群体的平均适应度在进化到第 100 左右刚好达到收敛,所以不是迭代次数的问题,就可能是种群规模的问题,就像在自然界进化过程当中,一个种群的规模越大,其产生优秀个体的可能性也就越大,经过进化后,就能产生更加优秀的群体。所以,我们要不断增加种群规模来比较其聚类效果,这里依次取 sizepop=200,300,400,500,…。当进化代数(即迭代次数)maxgen=100,种群规模 sizepop=700 时,聚类结果如图 8-4 所示,适应度曲线如图 8-5 所示。

但是当种群规模增加到 sizepop=700 左右时,聚类效果依然不佳,种群的平均适应度曲线并没有收敛,这时就需要增加进化代数 maxgen。这就像在自然界进化中,虽然一个种群的规模很大,产生优秀个体的可能性很大,但是没有经过长时间的进化,没有达到优胜劣汰的效果。

就这样,经过不断地增大种群规模,并找到其合适的进化代数来观察聚类的效果。但是,是不是种群规模越大,进化代数越大越好呢?显然不是的,种群规模越大,进化代数越大,聚类效果确实越好,但是付出的代价却是收敛速度越慢,所以我们要根据实际情况确定一个合适的种群规模和进化代数。

遗传算法实验的聚类结果如表 8-1 所示。

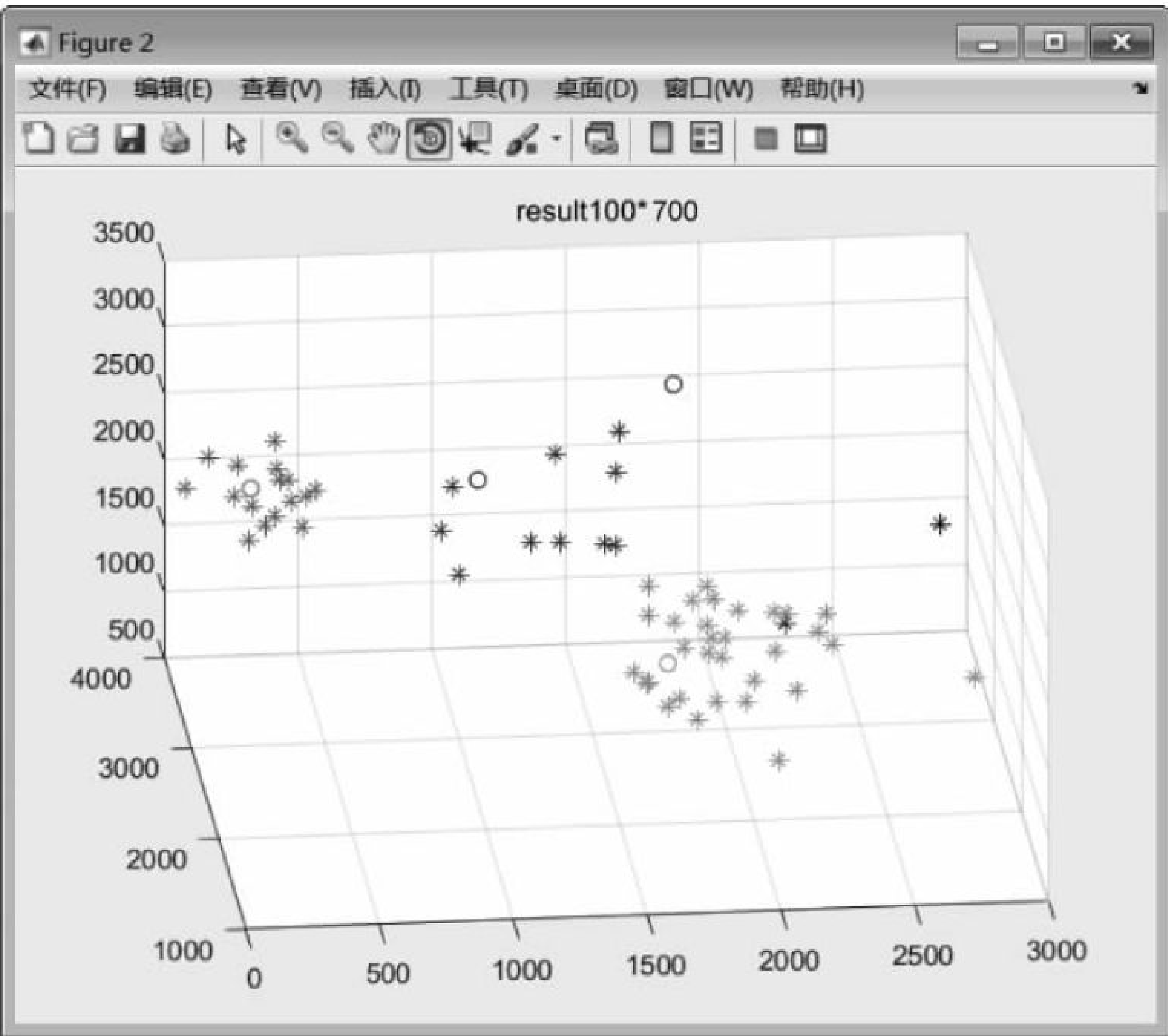


图 8-4 maxgen＝100，sizepop＝700 时的聚类结果

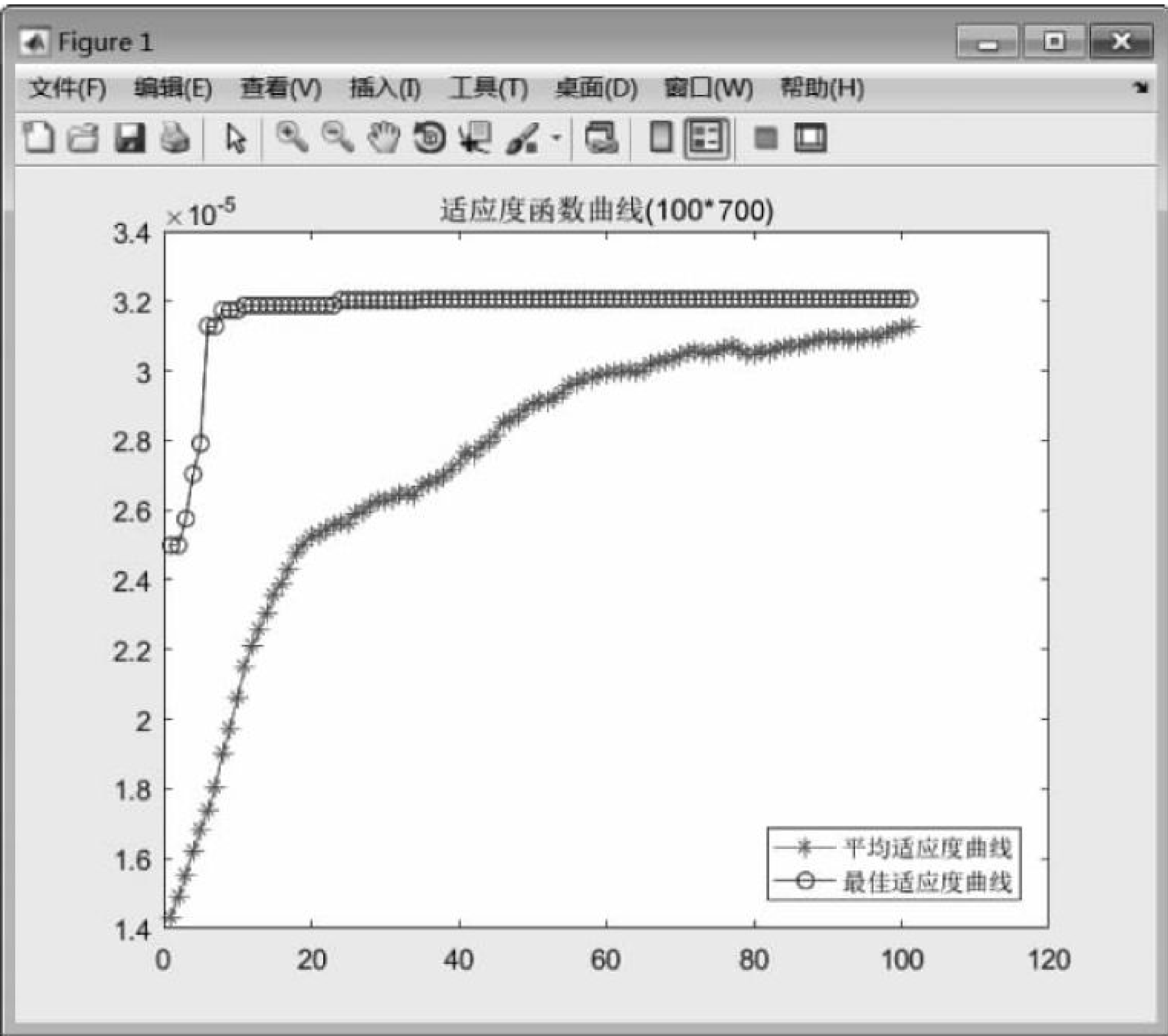


图 8-5 maxgen＝100，sizepop＝700 时的适应度函数曲线

表 8-1 聚类结果

种群规模	进化代数	运行次数	准则函数平均值	收敛速度平均值/s
100	100	5	42138	4.6325108
200	100	5	49595	8.4666738
300	100	5	43696	12.320991
400	100	5	40945	17.223390
500	100	5	40555	19.819010
600	100	5	43832	24.137784
700	100	5	44060	29.798188
800	150	5	47935	46.834691
900	150	5	36118	54.178806
1000	150	5	37477	63.526360
1100	150	5	44204	65.906776
1200	150	5	31969	68.900633
1500	200	5	30493	114.950724
2000	200	5	33498	153.360266
2500	250	5	37530	234.189298
3000	300	5	40662	332.896752
4000	300	5	39092	446.044975

通过对比表中数据可以看出,随着种群规模和进化代数的增加,准则函数的值得到明显下降,得出了正确的聚类结果,但是具有很大的随机性,收敛速度也越来越慢。当进化代数(即迭代次数)maxgen=200,种群规模 sizepop=1500 时,准则函数平均值最小,聚类结果是实验中最好的,聚类结果如图 8-6 所示,适应度函数曲线如图 8-7 所示。

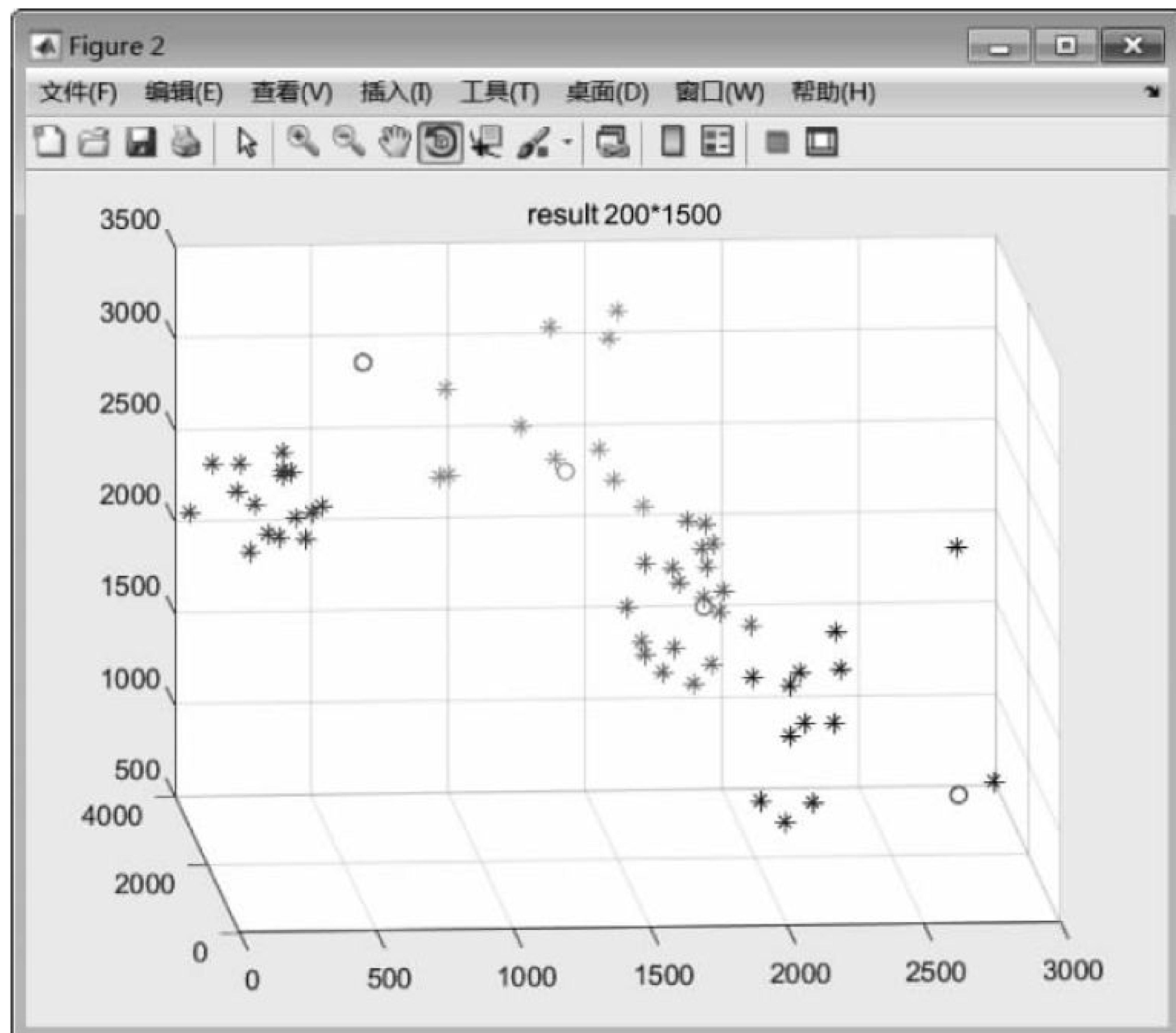


图 8-6 maxgen=200,sizepop=1500 时的聚类结果

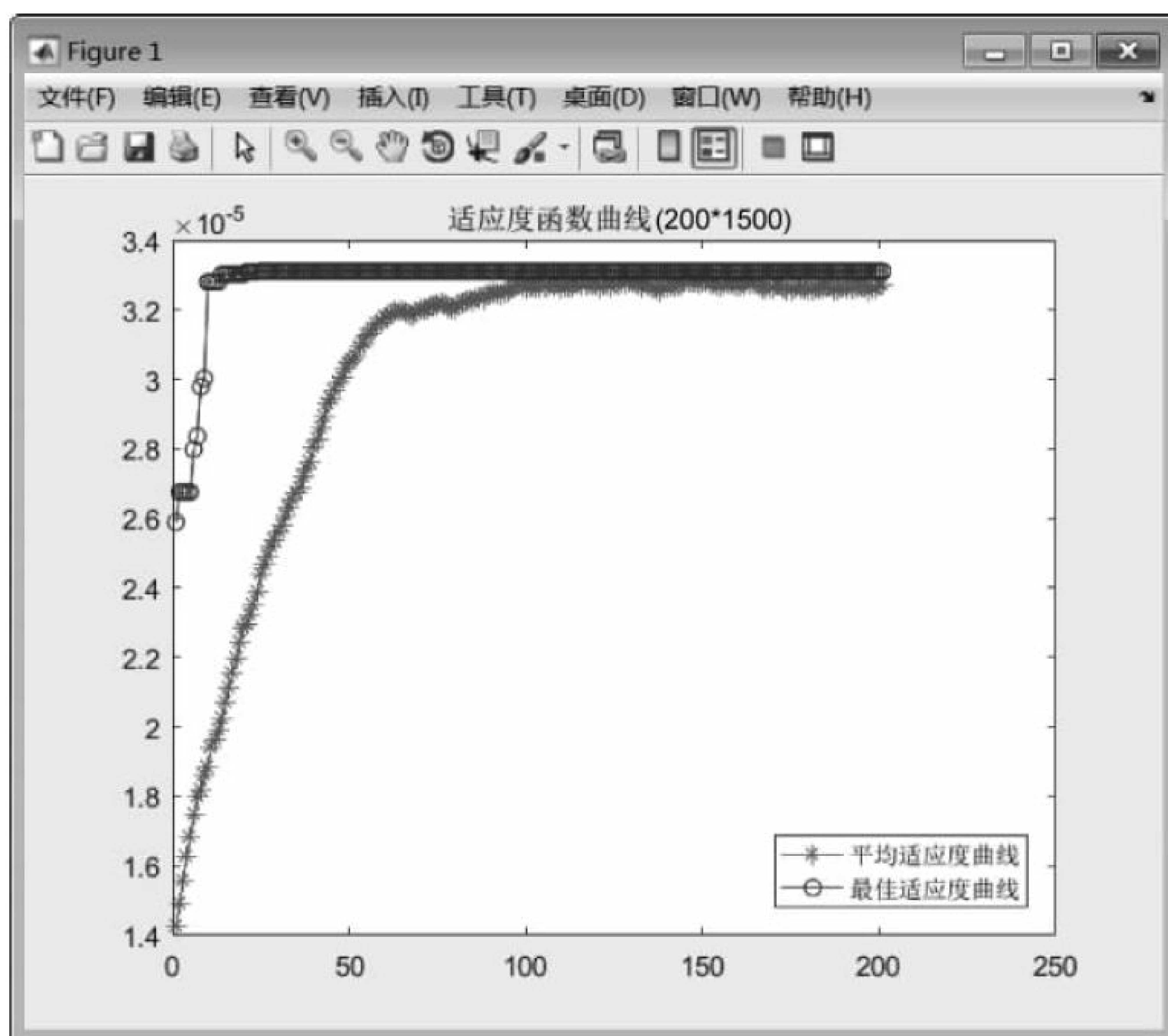


图 8-7 maxgen=200,sizepop=1500 时的适应度函数曲线

8.4 结论

本章给出了遗传算法的概念、原理、特点及实现的流程,通过实例介绍了遗传算法在聚类问题的实现方法与步骤。

习题

(1) 什么是遗传算法?
(2) 常用的遗传算子有哪些?
(3) 遗传算法的特点是什么?
(4) 简述遗传算法的基本要素。

第9章

蚁群算法聚类设计

蚁群优化算法是由意大利学者 M. Dorigo 等人提出的一种新型的解决组合优化问题的模拟进化算法。该算法不仅能够实现智能搜索、全局优化，而且具有稳健性、正反馈、分布式计算、易与其他算法相结合等特点。

9.1 蚁群算法简介

蚁群算法最初由意大利学者 M. Dorigo 等人于 1991 年首次提出。根据蚂蚁“寻找食物”的群体行为，M. Dorigo 在会议 ECAL(European Conference on Artificial Life)上最早提出了蚁群算法的基本模型，1992 年 M. Dorigo 又在其博士学位论文中进一步阐述了蚁群算法的核心思想。在 1996 年，M. Dorigo 又一篇奠基性文章 *Ant system：optimization by a colony of cooperation agents* 在 *IEEE Transactions on Systems，Man，and Cybernetics—PartB* 上发表，在该文中，M. Dorigo 等不仅对蚁群算法的基本原理和数学模型做了更加系统的阐述，还将其与遗传算法、禁忌搜索算法、模拟退火算法、爬山法等进行了仿真实验比较，并把算法拓展到解决非对称旅行商问题(travelling sales man problem，TSP)、指派问题(quadratic assignment problem，QAP)以及车间作业调度问题(job-shop scheduling problem，JSP)，并且对算法中初始参数对性能的影响做了初步探讨。自 1996 年起，蚁群算法作为一种新颖的、处于前沿的问题优化求解算法，在算法的改进、算法收敛性的证明以及应用领域方面逐渐得到了世界许多国家研究者的关注。进入 21 世纪，国际著名的顶级学术刊物 *Nature* 多次报道了蚁群算法的研究成果，*Future Generation Computer Systems* 和 *IEEE Transaction on Evolutionary Computation* 分别出版了蚁群算法专刊。目前，对蚁群算法及其应用的研究已经成为国内外许多学术期刊和会议上的一个研究热点和前沿性课题。随着蚁群算法研究的兴起，人们发现在某些方面采用蚁群模型进行聚类更加接近实际聚类问题。

蚁群优化算法是一种新型的解决组合优化问题的模拟进化算法。它是模拟自然界中蚂蚁的觅食行为产生的。蚂蚁在运动过程中不仅能够在所经路径上留下一种叫作信息素

(pheromone)的物质,而且它们还能够感知到这种物质的存在,并以此指导自己的运动方向。蚂蚁个体之间通过这种信息交流达到搜索食物的目的。利用正反馈原理,可以加快进化过程;分布式计算使该算法易于并行实现,个体之间不断进行信息交流和传递,有利于找到较好的解;该算法易与多种启发式算法结合,可改善算法的性能;由于健壮性强,故在基本蚁群算法模型的基础上进行改进,便可用于其他问题。因此,蚁群算法的问世为诸多领域解决复杂优化问题提供了有力的工具。

9.2　蚁群算法原理

9.2.1　基本蚁群算法原理

现实生活中单个蚂蚁的能力和智力非常简单,但在蚂蚁寻找食物的过程中,往往能找到蚁穴与食物之间的最佳行进路线。不仅如此,蚂蚁还能够适应环境变化。例如,在蚂蚁运动路线上突然出现障碍物时,一开始蚂蚁分布是均匀的,不管路径长短,蚂蚁总是先按照同等概率选择各条路径,如图9-1所示。但经过一段时间后,蚂蚁能够很快重新找到最优路径。

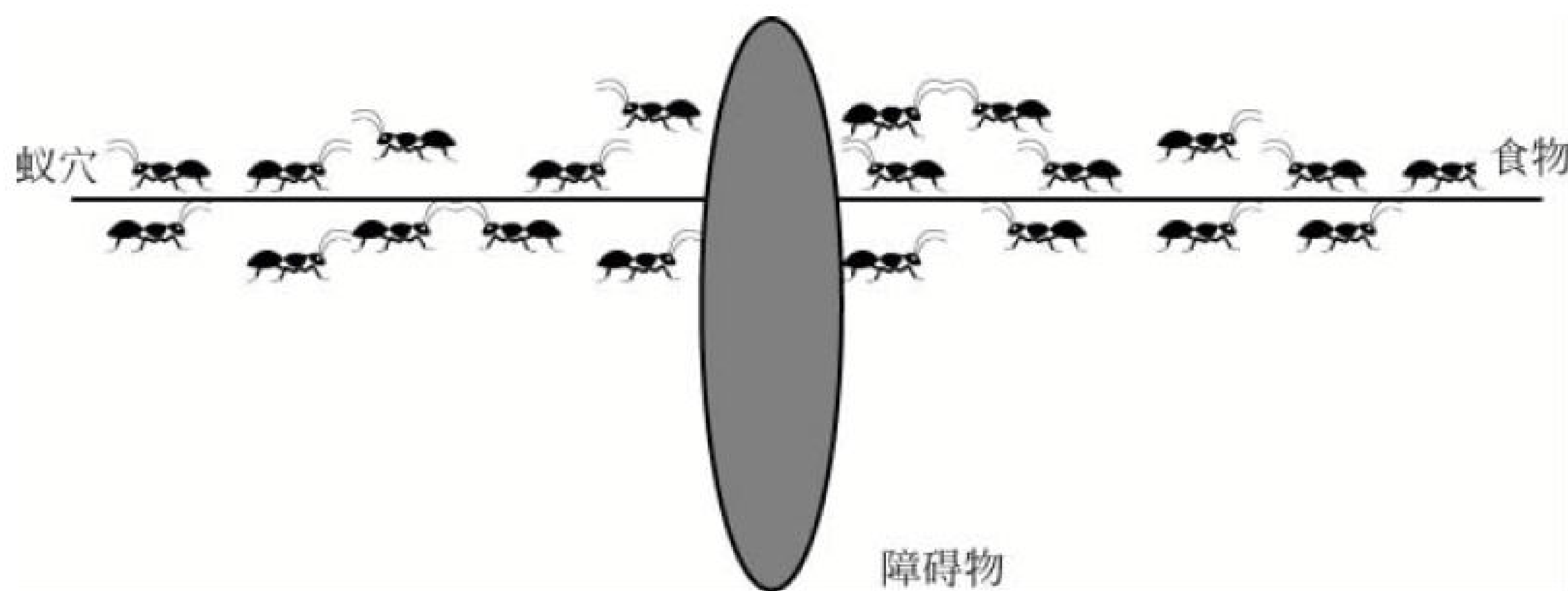


图9-1　蚂蚁以等同概率选择各条路径

蚁群的这些特性早就引起了生物学家和仿生学家的强烈兴趣。仿生学家通过大量的细致观察和研究发现,蚂蚁个体之间通过信息素进行信息传递,从而相互协作,完成复杂的任务。蚁群之所以表现出复杂有序的行为,个体之间的信息交流与相互协作起着重要的作用。

蚂蚁在运动过程中,能够在其所经过的路径上留下信息素,而且蚂蚁在运动过程中能够感知到信息素的存在,并以此确定自己的运动方向。蚂蚁倾向于朝着该物质强度高的方向移动。如果路径上出现障碍物时,相等时间内蚂蚁留在较短路径上的信息素比较多,这样就形成了正反馈现象,选择较短路径的蚂蚁也会随之增多,如图9-2所示。

蚂蚁运动过程中,较短路径上遗留的信息素会在很短时间内大于较长路径的信息素,原因不妨用图9-3说明:假设A、E两点分别是蚁群的巢穴和食物源,之间有两条路径A—B—H—D—E和A—B—C—D—E,其中B—H和H—D间距离为1m,B—C和C—D间距离为0.5m。

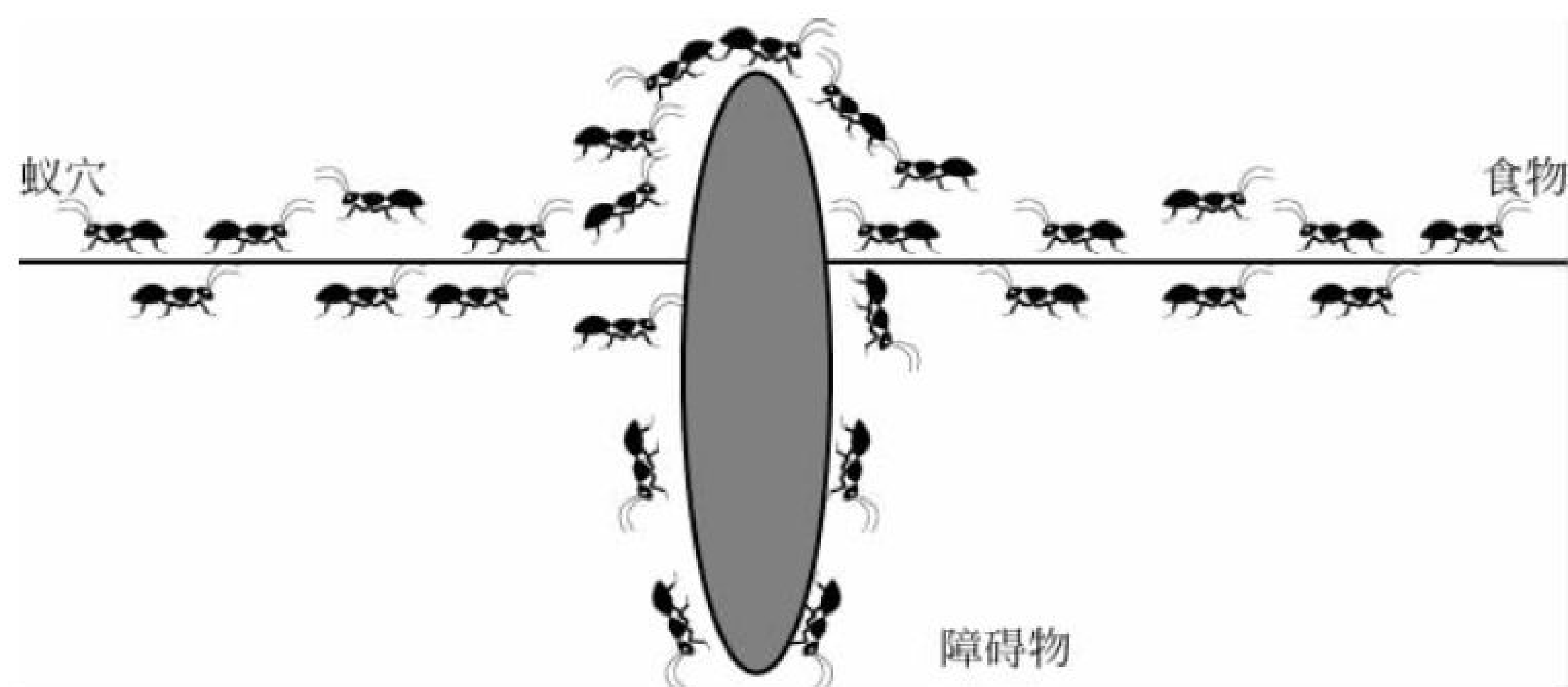


图 9-2 较短路径信息素浓度高,选择该路径的蚂蚁增多

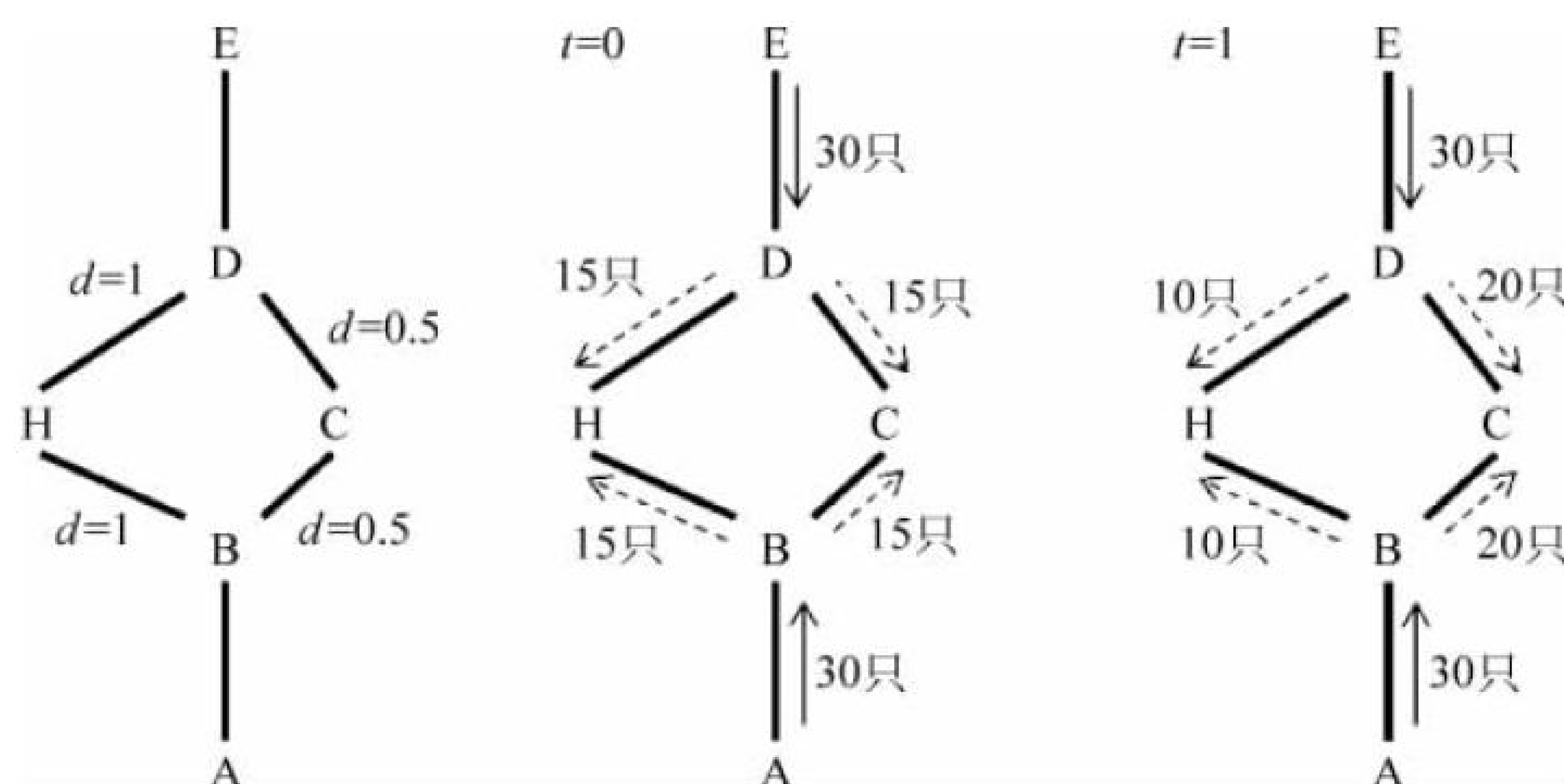


图 9-3 蚂蚁选择路径示意图

最初(即 $t=0$ 时刻),如图 9-3 所示,当 30 只蚂蚁走到分支路口 B 或者 D 点时,要决定往哪个方向走。因为初始时没有什么线索可以为蚂蚁提供选择路径的标准,所以它们就以相同的概率选择路径,结果有 15 只蚂蚁走左边的路径 D—H、B—H,另外 15 只蚂蚁走右边的路径 D—C、B—C,这些蚂蚁在行进过程中分别留下信息素。假设蚂蚁都具有相同的速度(1m/s)和信息素释放能力,则经过 1s 后从 D 点出发的 30 只蚂蚁有 15 只到达了 H 点,有 15 只经过 C 点到达了 B 点;同样,从 B 点出发的 30 只蚂蚁有 15 只到达了 H 点,有 15 只经过 C 点到达了 D 点。很显然,在相等的时间间隔内,路径 D—H—B 上共有 15 只蚂蚁经过并遗留了信息素,D—C—B 上却有 30 只蚂蚁经过并遗留了信息素,其信息素浓度是 D—H—B 路径上的 2 倍。因此,当 30 只蚂蚁分别回到 A、E 点重新选择路径时就会以 2 倍于 D—H—B 的概率选择路径 D—C—B,从而使 D—H—B 上的蚂蚁数目变成了 10 只,距离较短的路径上信息素很快得到了强化,其优势也很快被蚂蚁发现。

不难看出,由大量蚂蚁组成的群体的集体行为表现出了一种信息正反馈现象:某条路径上走过的蚂蚁越多,则后来者选择该路径的概率就越大。蚂蚁个体之间就是通过这种信息的交流来达到搜索食物的目的,并最终沿着最短路径行进,如图 9-4 所示。

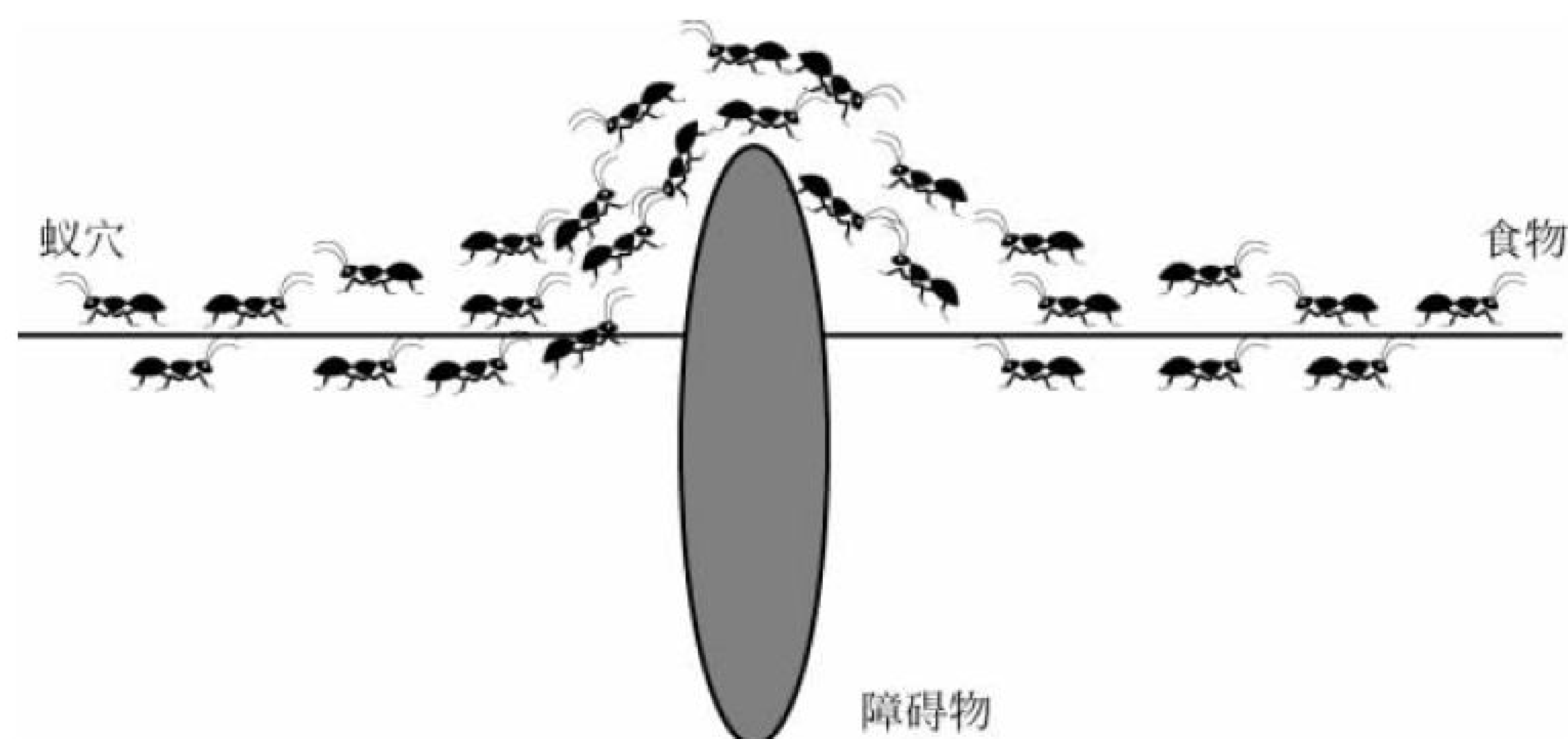


图 9-4 蚂蚁最终找到绕过障碍物的最优路径

9.2.2 模型建立

1. 基于蚂蚁构造墓地和分类幼体的聚类分析模型

蚁群构造墓地行为和分类幼体行为统称为蚁群聚类行为。生物学家经过长期的观察发现，在蚂蚁群体中存在一种本能的聚集行为。蚂蚁往往能在没有关于蚂蚁整体的任何指导性信息的情况下，将其死去同伴的尸体安放在一个固定的场所。Chretien 用 Lasiusniger 蚂蚁做了大量试验，研究蚂蚁的这种构造墓地行为，发现工蚁能在几小时内将分散在蚁穴内各处的任意分布、大小不同的蚂蚁尸体聚成几类；J. L. Deneubourg 等人也用 Pheidole pallidula 蚂蚁做了类似的实验。另外观察还发现，蚁群会根据蚂蚁幼体的大小，分别把其堆放在蚁穴周围和中央的位置。真实的蚁群聚类行为的实验结果如图 9-5 所示，四张照片分别对应实验初始状态、3 小时、6 小时和 36 小时的蚁群聚类情况。这种蚁群聚集现象的基本机制是小的聚类通过已聚集的蚂蚁尸体发出的信息素来吸引工蚁存放更多的同类对象，由此变成更大的聚类。在这种情况下，蚁穴环境中的聚类分布特性起到了间接通信的作用。

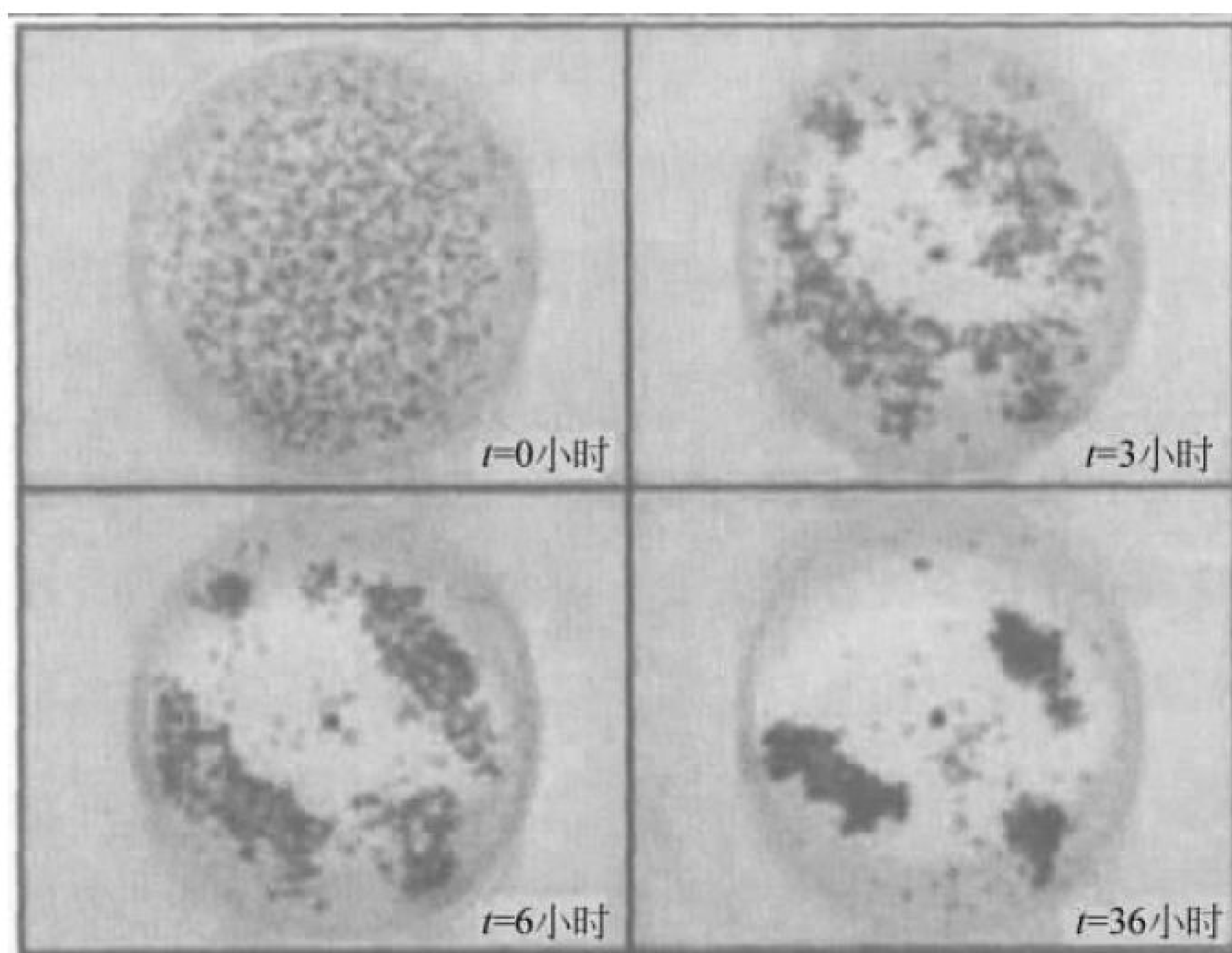


图 9-5 真实蚁群的聚类行为

针对蚂蚁构造墓地和分类幼体的本能所表现出来的聚类行为，Deneubourg 等人提出了蚁群聚类的基本模型(BM)用来解释这种现象，指出单个的对象会比较容易被拾起并且移动到其他具有很多这类对象的地方。

基本模型经过利用个体与个体和个体与环境之间的交互作用，实现了自组织聚类，并成功地应用于机器人的控制中(一群类似于蚂蚁的机器人在二维网格中随意移动并可以搬运基本物体，最终把它们聚集在一起)。该模型成功地应用引起了各国学者的广泛关注和研究热潮。E. Lumer 和 B. Faieta 通过在 Deneubourg 的基本分类模型中引入数据对象之间相似度的概念，提出了 LF 聚类分析算法，并成功地将其应用到数据分析中。

2. 基于蚂蚁觅食行为和信息素的聚类分析模型

蚂蚁在觅食过程中，能够分为搜索食物和搬运食物两个环节。每只蚂蚁在运动过程中都将会在其所经过的路径上留下信息素，而且能够感知到信息素的存在及其强度，比较倾向于向信息素强度高的方向移动。同样信息素自身也会随着时间的流逝而挥发，显然某一路径上经过的蚂蚁数目越多，那么其信息素就越强，以后的蚂蚁选择该路径的可能性就比较大，整个蚁群的行为表现出了信息正反馈现象。

通过借鉴这一蚁群生态原理，基于蚂蚁觅食行为和信息素的聚类分析模型的基本思想是将数据看作是具有不同属性的蚂蚁，聚类中心就被视为蚂蚁所要寻找的“食物源”，数据聚类过程可以看作是蚂蚁进行找寻食物源的过程。该模型的算法流程如图 9-6 所示。

该模型可以描述为：假设待分类的数据对象有 N 个，每个数据对象有 m 个属性，数据对象定义为 $X=\{X_i \mid X_i=(x_{i1},x_{i2},\cdots,x_{im}),i=1,2,\cdots,N\}$，分类数目是 K 类。在模式样本 i 处分别放置一个蚂蚁，模式样本 i 分配给第 j 个聚类中心 $C_j(j=1,2,\cdots,K)$，蚂蚁就在模式样本 i 到聚类中心 C_j 的路径(i,j)上留下信息素 $\tau_{ij}(t)$。$d(X_i,C_j)$表示 X_i 到聚类中心 C_j 之间的欧氏距离；$P_{ij}(t)$是蚂蚁选择路径(i,j)的概率，计算公式为

$$P_{ij}(t)=\frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum\limits_{s\in S}\tau_{sj}^{\alpha}(t)\eta_{sj}^{\beta}(t)},\quad i=1,2,\cdots,N;\ j=1,2,\cdots,K \tag{9-1}$$

$$\tau_{ij}(t)=\begin{cases}1, & d(X_i,C_j)\leqslant R\\ 0, & d(X_i,C_j)>R\end{cases} \tag{9-2}$$

$$d(X_i,C_j)=\sqrt{\sum_{r=1}^{m}(x_{ir}-c_{jr})^2} \tag{9-3}$$

其中，R 是聚类半径；$S=\{s \mid d(X_s,C_j)\leqslant R,s=1,2,\cdots,N$ 且 $s\neq j\}$表示分布在聚类中心 C_j 邻域内的数据对象的集合；$\eta_{ij}(t)=1/d(X_i,C_j)$表示 t 时刻模式样本 i 分配给第 j 个聚类中心 C_j 的启发信息数值；α 和 β 是用于控制信息素和启发数的可调节参数；如果 $P_{ij}(t)$大于阈值 P_0，就将 X_i 归并到 C_j 的邻域。

模型终止条件是所有聚类总偏离误差 ξ 小于给定的统计误差 ε_0。所有聚类的总偏离误差 ξ 计算公式为

$$\xi=\sum_{j=1}^{K}\xi_j,\quad j=1,2,\cdots,K(K\text{ 表示总类别数}) \tag{9-4}$$

$$\xi_j=\sqrt{\frac{1}{J}\sum_{i=1}^{J}(X_i-C_j')^2} \tag{9-5}$$

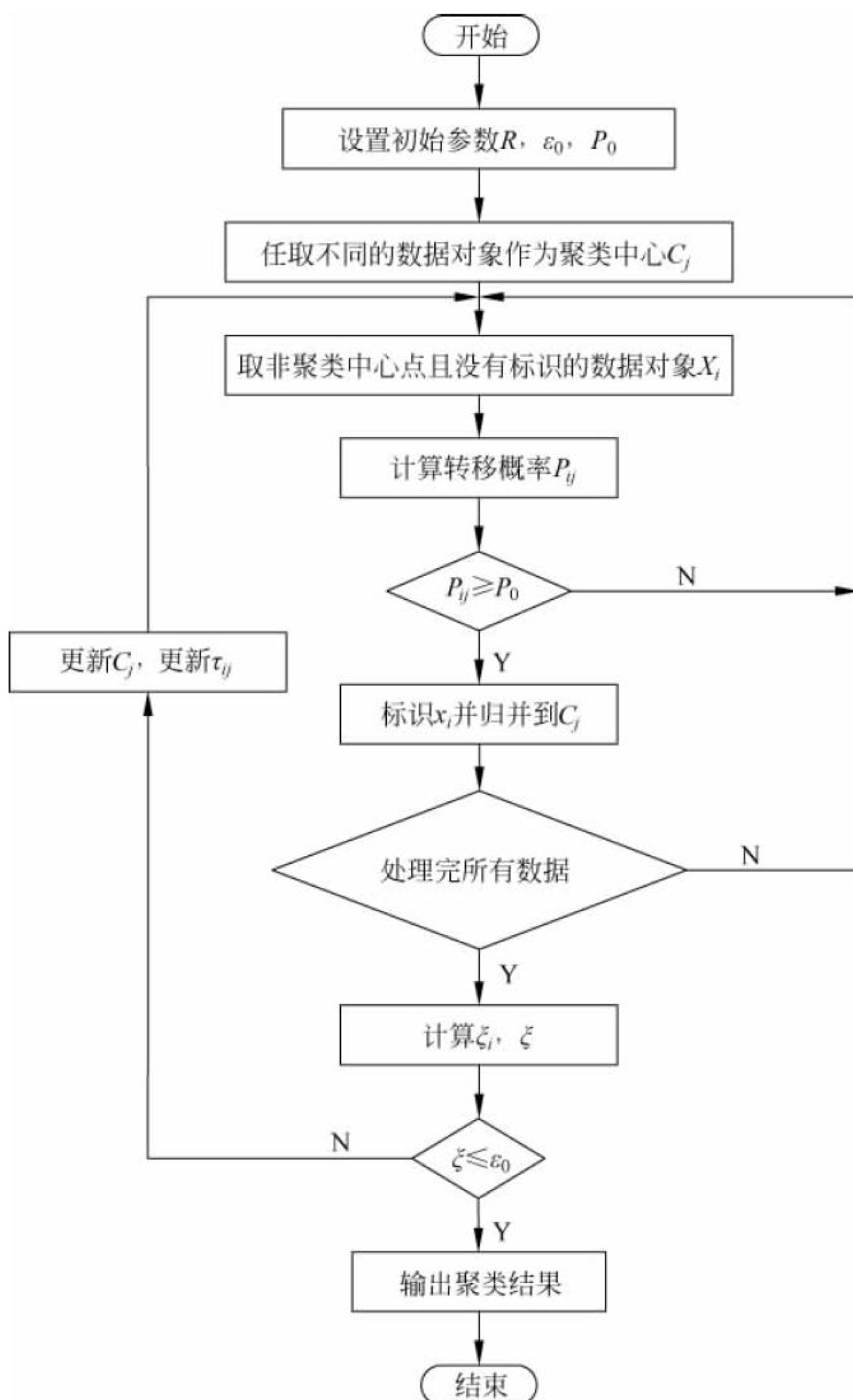


图 9-6 基于蚂蚁觅食行为和信息素的聚类分析模型流程图

$$C_j' = \frac{1}{J}\sum_{i=1}^{J} X_i \tag{9-6}$$

其中 ξ_j 表示第 j 个聚类的偏离误差，C_j'为新的聚类中心，X_i 是所有归并到 C_j'类中的数据对象，即 $X_i \in \{X_h \mid d(X_h, C_j') \leqslant R;\ h=1,2,\cdots,N\}$，$J$ 为该聚类中所有数据对象的个数。

该模型中蚂蚁的通信介质是其在路径上留下的信息素，具有自组织、正反馈等优点。尽管该方法不需要事先给定聚类的个数，但由于需要预先设置类半径，因此限制了生成的类的规模。而且由于信息素的更新原则 $\tau_{ij}(t)$取为常数 1，处理策略使用的是局部信息，对数据关联性也没有考虑到，所以非常容易陷入局部最优。通过引入蚁群算法的 Ant-Cycle 模型中信息素的处理方式，$\tau_{ij}(t)$为在本次循环中所经过的路径总长度的函数，能更加充分地利用环境中的整体信息。这种信息的更新规则能够让短路径上对应的信息量逐渐增大，这样

就充分体现出算法中全局范围内比较短的路径的生存能力,加强了信息的正反馈性能,而且提高算法系统搜索收敛的速度。同时,能更好地保证残余信息伴随着时间的推移而逐渐减弱,把不好的路径“忘记”,这样即使路径常常被访问也不至于因为信息素积累而使得期望值的作用没法体现出来。

9.2.3 蚁群算法的特点

蚁群算法的主要特点是通过正反馈、分布式协作来寻找最优解,这是一种基于种群寻优的启发式搜索算法,能根据聚类中心的信息量把周围数据归并到一起,从而得到聚类分类。其具体步骤为:变量初始化;将 m 只蚂蚁放到 n 个城市中;m 只蚂蚁按照概率函数选择下一座城市,完成各自的周游;记录本次迭代的最佳路线;更新信息素;禁忌表清零;输出结果。

蚁群算法来源于蚂蚁搜索食物的过程,与其他群集智能一样,具有较强的健壮性,不会由于某一个或者某几个个体的故障而影响整个问题的求解,具有良好的可扩充性,由系统中个体的增加而增加的系统通信开销非常小。

除此之外,蚁群系统还具有以下特点:

(1) 蚁群算法是一种并行的优化算法。蚂蚁搜索食物的过程彼此独立,只通过信息素进行间接的交流。这为并行计算旅行商问题提供了极大的方便。旅行商问题的计算量一般较大,使用并行计算可以显著减少计算时间。

(2) 蚁群算法是一种正反馈算法。一段路径上的信息素水平越高,就越能够吸引更多的蚂蚁沿着这条路径运动,这又使得其信息素水平增加。正反馈的存在使得搜索很快收敛。

(3) 蚁群算法的健壮性较好。相对于其他算法,蚁群算法对初始路线的要求不高。也就是说,蚁群算法的搜索结果不依赖于初始路线的选择。

(4) 蚁群算法的搜索过程不需要进行人工调整。相对于某些需要进行人工干预的算法(如模拟退火算法),蚁群算法可以在不需要人工干预的情况下完成从初始化到得到整个结果的全部计算过程。

蚁群算法对于小规模(不超过30)的旅行商问题效果显著,但对于较为复杂的旅行商问题,其性能急剧下降。主要原因是,在该算法的初始阶段,各条路径上的信息素水平基本相等,蚂蚁的搜索呈现出较大的盲目性。只有经过较长时间后,信息素水平才呈现出明显的指导作用。另外,由于蚁群算法是一种正反馈算法,在算法速度收敛较快的同时,也容易陷入局部优化。比如说,在两个旅行点中间的一条边,这条边的旅行费用在所有相邻的城市中是最低的,那么,在搜索的初期,这条边上会获得最高的信息素水平。高信息素水平又容易导致更多的蚂蚁沿这条路径运动,这样与这两个城市相连的路径就没有太多的机会被访问,但实际上,全局最优路径中,并不一定包含这条边。因此对于大规模的旅行商问题,早期的蚁群算法搜索到最优解的可能性较小。

另外,蚁群算法仍然存在一些缺陷,如在性能方面,算法的收敛速度和所得解的多样性、稳定性等性能之间存在矛盾。这是因为蚁群中多个个体的运动是随机的,虽然通过信息交流能够向着最优路径进化,但是当群体规模较大时,很难在较短时间内从杂乱无章的路径中找到一条较好的路径。如果加快收敛速度则很可能导致蚂蚁的搜索陷入局部最优,造成早

熟、停滞现象。

在应用范围方面，蚁群算法的应用局限在较小的范围内，难以处理连续空间的优化问题。由于每只蚂蚁在每个阶段所做的选择总是有限的，它要求离散的解空间，因而对组合优化等离散问题很适用，而对线性和非线性规划等连续空间的优化问题的求解不能直接应用。

9.3 基本蚁群算法的实现

本例使用表 1-1 中的三元色数据，按照颜色数据所表征的特点，将样本按照各自所属的类别归类。

由于蚁群优化算法是迭代求取最优值，所以事先无须训练数据，故取 59 组数据确定类别。下面使用 MATLAB 构建蚁群优化算法。程序流程如图 9-7 所示。

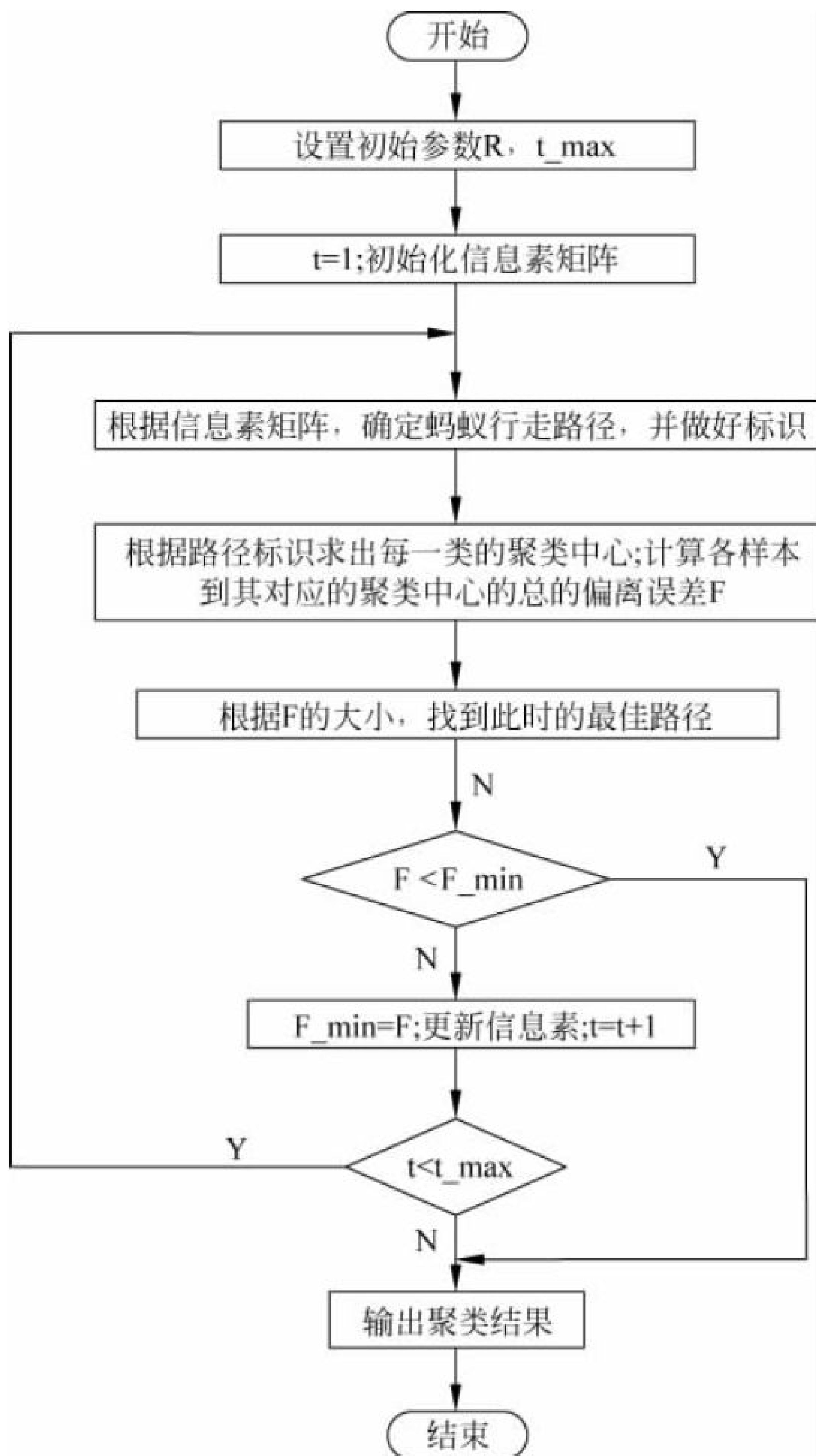


图 9-7 程序流程图

1. 程序初始化

加载测试样本矩阵 X,根据测试样本,给出样本个数 N,测试样本的属性数 n(即维数),给定聚类个数 K(即要分成几类),给定蚂蚁数 R,最大迭代次数 t_max,最佳路径的偏差值 best_solution_function_value,初始值设置为无穷大。

初始化程序代码如下:

```
X = load('data.txt');
[N,n] = size(X);                  % N = 测试样本数;n = 测试样本的属性数
K = 4;                            % K = 组数
R = 100;                          % R = 蚂蚁数
t_max = 1000;                     % t_max = 最大迭代次数
best_solution_function_value = inf;   % 最佳路径度量值(初值为无穷大,该值越小聚类效果越好)
```

2. 信息素矩阵初始化

信息素矩阵维数为 N * K(样本数 * 聚类数)初始值为 0.01。

信息素矩阵初始化程序如下:

```
c = 10 ^ - 2;
tau = ones(N,K) * c;          % 信息素矩阵,初始值为 0.01 的 N * K 矩阵(样本数 * 聚类数)
```

3. 蚂蚁路径的选择及标识

定义标识字符矩阵 solution_string,维数为 R * N+1,初始值都为 0,以信息矩阵中信息素的值确定路径(即确定分到哪一组),具体方法如下:

如果该样本各信息素的值都小于信息素阈值 q,则取信息素最大的作为路径。若最大值有多个,则从相同的最大值中随机取一个作为路径。

若信息素大于阈值 q,则求出各路径信息素占该样本总信息素的比例,以概率确定路径。

4. 聚类中心选择

聚类中心为该类所有样本的各属性值的平均值。

5. 偏离误差计算

偏离误差的计算,即各样本到其对应的聚类中心的欧氏距离之和 F。F 越小,聚类效果越好。计算每只蚂蚁的 F 值,找到最小的 F 值,该值对应的路径为本次迭代的最佳路径。

6. 信息素更新

对信息素矩阵进行更新,更新方法为:新值为原信息素值乘以(1−rho),rho 为信息素蒸发率,再加上最小偏差值的倒数。

程序代码如下：

```
for i = 1 : N
        tau(i,best_solution(1,i)) = (1 - rho) * tau(i,best_solution(1,i)) + 1/ tau_F;
```

信息素更新之后，再根据新的信息素矩阵判断路径，进行迭代运算，直到达到最大迭代次数，或偏离误差达到要求值。

7. 完整程序及仿真结果

蚁群优化算法的完整 MATLAB 程序如下：

```
clc;
clf;
clear;
%X = 测试样本矩阵;
X = load('data.txt');
[N,n] = size(X);                %N = 测试样本数;n = 测试样本的属性数
K = 4;                          %K = 组数
R = 100;                        %R = 蚂蚁数
t_max = 1000;                   %t_max = 最大迭代次数
%初始化
c = 10 ^ - 2;
tau = ones(N,K) * c;            %信息素矩阵,初始值为 0.01 的 N * K 矩阵(样本数 * 聚类数)
q = 0.9;                        %阈值 q
rho = 0.1;                      %蒸发率
best_solution_function_value = inf; %最佳路径度量值(初值为无穷大,该值越小聚类效果越好)
tic
t = 1;
while ((t <= t_max))            %进行 t_max 次迭代计算
    solution_string = zeros(R,N + 1);          %路径标识字符:标识每只蚂蚁的路径
     for i = 1 : R              %以信息素为依据确定蚂蚁的路径
         r = rand(1,N);         %随机产生值为 0～1 随机数的 1 * 51 的数组
         for g = 1 : N
              if r(g) < q       %如果 r(g)小于阈值
                  tau_max = max(tau(g,:));
                  Cluster_number = find(tau(g,:) == tau_max);
                  %确定第 i 只蚂蚁对第 g 个样本的路径标识
                  solution_string(i,g) = Cluster_number(1);
                  %如果 r(g)大于阈值,求出各路径信息素占总信息素的比例,按概率选择
                  %路径
                else
                   sum_p = sum(tau(g,:));
                   p = tau(g,:) / sum_p;
                   for u = 2 : K
                        p(u) = p(u) + p(u - 1);
                   end
                  rr = rand;
```

```
                    for s = 1 : K
                        if (rr <= p(s))
                            Cluster_number = s;
                            solution_string(i,g) = Cluster_number;
                        break;
                    end
                end
            end
        end
        % 计算聚类中心
        weight = zeros(N,K);
          for h = 1:N                                         % 给路径做计算标识
              Cluster_index = solution_string(i,h);           % 类的索引编号
                weight(h,Cluster_index) = 1;           % 对样本选择的类在 weight 数组的相应
                                                       % 位置标 1
          end
              cluster_center = zeros(K,n);                     % 聚类中心(聚类数 K 个中心)
              for j = 1:K
                  for v = 1:n
                      sum_wx = sum(weight(:,j).*X(:,v)); % 各类样本各属性值之和
                      sum_w = sum(weight(:,j));             % 各类样本个数
                      if sum_w == 0            % 该类样本数为 0,则该类的聚类中心为 0
                        cluster_center(j,v) = 0
                         continue;
                  else          % 该类样本数不为 0,则聚类中心的值取样本属性值的平均值
                  cluster_center(j,v) = sum_wx/sum_w;
                  end
              end
          end
% 计算各样本点各属性到其对应的聚类中心的均方差之和,该值存入 solution_string 的最后一位
          F = 0;
          for j = 1:K
              for ii = 1:N
                  Temp = 0;
                  if solution_string(i,ii) == j;
                      for v = 1:n
                          Temp = ((abs(X(ii,v) - cluster_center(j,v))).^2) + Temp;
                      end
                      Temp = sqrt(Temp);
                  end
                F = (Temp) + F;
              end
          end
           solution_string(i,end) = F;
    end
    % 根据 F 值,把 solution_string 矩阵升序排序
    [fitness_ascend,solution_index] = sort(solution_string(:,end),1);
    solution_ascend = [solution_string(solution_index,1:end-1) fitness_ascend];
   for k = 1:R
            if solution_ascend(k,end)<= best_solution_function_value
```

```
                best_solution = solution_ascend(k,:);
            end
        k = k+1;
        end
    %用最好的L条路径更新信息素矩阵
    tau_F = 0;
    L=2;
    for j = 1:L
      tau_F = tau_F + solution_ascend(j,end);
    end
    for i = 1 : N
      tau(i,best_solution(1,i)) = (1 - rho) * tau(i,best_solution(1,i)) + 1/ tau_F;
    %1/tau_F 和 rho/tau_F 效果都很好
    end
    t=t+1;
end
time=toc;
clc
t
time
cluster_center
best_solution = solution_ascend(1,1:end-1);
IDY=ctranspose(best_solution)
best_solution_function_value = solution_ascend(1,end)
%分类结果显示
plot3(cluster_center(:,1),cluster_center(:,2),cluster_center(:,3),'o');grid;box
title('蚁群聚类结果(R=100,t=10000)')
xlabel('X')
ylabel('Y')
zlabel('Z')
YY=[1 2 3 4];
index1 = find(YY(1) == best_solution)
index2 = find(YY(2) == best_solution)
index3 = find(YY(3) == best_solution)
index4 = find(YY(4) == best_solution)
line(X(index1,1),X(index1,2),X(index1,3),'linestyle','none','marker','*','color','g');
line(X(index2,1),X(index2,2),X(index2,3),'linestyle','none','marker','*','color','r');
line(X(index3,1),X(index3,2),X(index3,3),'linestyle','none','marker','+','color','b');
line(X(index4,1),X(index4,2),X(index4,3),'linestyle','none','marker','s','color','b');
rotate3d
```

程序运行完以后,聚类结果如图9-8所示。从该图中可以看出基本蚁群聚类法的分类效果不太好。

程序运行结果:

```
t =
1001
time =
   23.4018
```

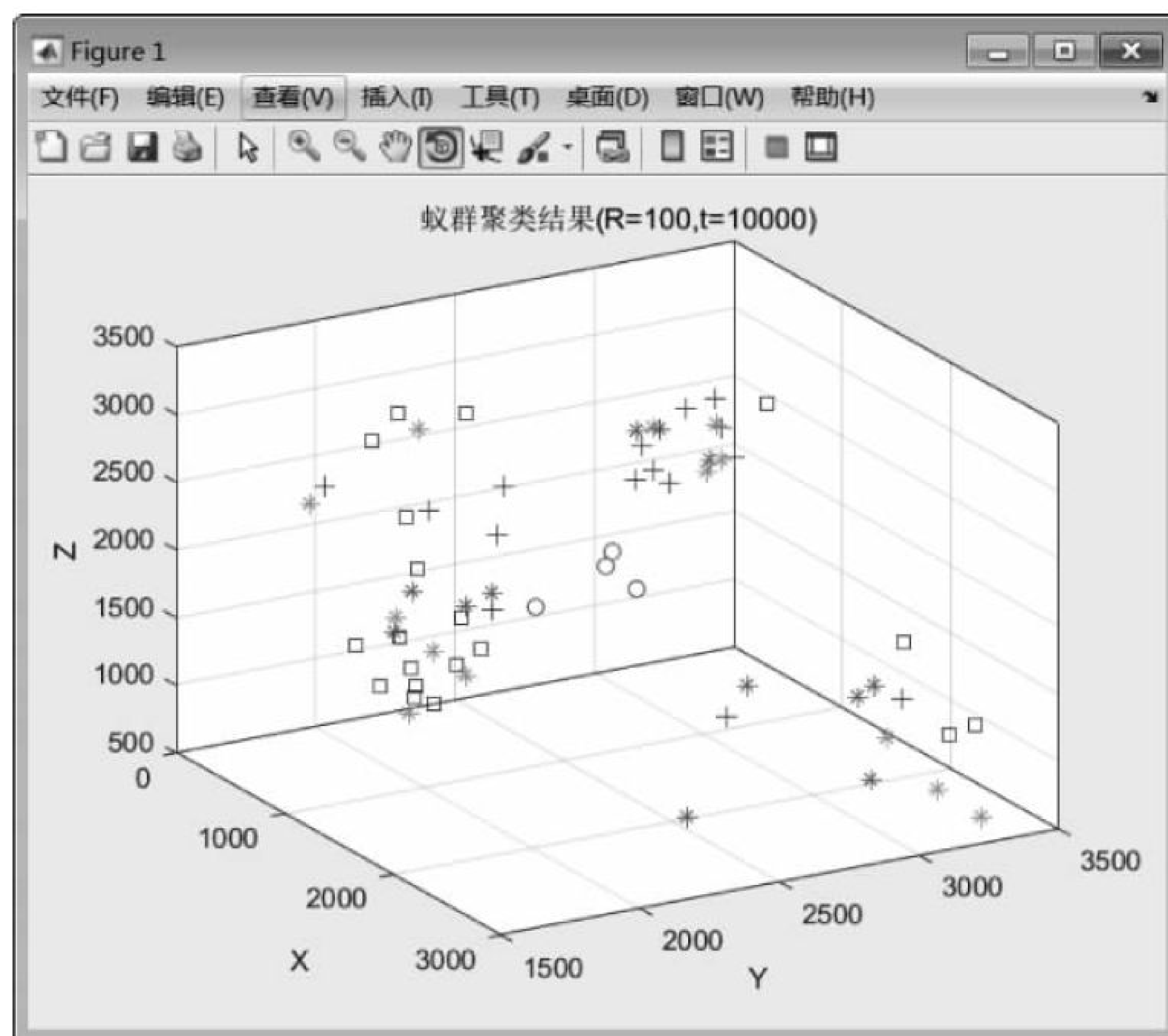


图 9-8 蚁群聚类结果

```
cluster_center =
   1.0e+03 *
    1.3710    2.6187    1.8872
    1.3950    2.4997    2.1124
    1.1438    2.6196    2.0613
    1.6024    2.1673    2.0350
best_solution_function_value =
   6.3409e+04
index1 =
3    4    6    14    19    27    34    37    41    44    48    49    57
index2 =
1    2    7    9    15    23    24    40    43    45    50    58
index3 =
5    8    12    13    17    18    28    29    32    38    39    46    54
55    56
index4 =
1～15 列
10    11    16    20    21    22    25    26    30    31    33    35    36
42    47
16～19 列
  52    53    59
```

9.4 算法改进

9.4.1 MMAS 算法简介

最大-最小蚂蚁系统(MAX-MIN ant system,MMAS)是在基本蚂蚁算法的基础上提出来的一种改进的蚁群算法。MMAS 将各条路径上的信息素初始值设为最大,并且规定了各条路径上的最小信息素的值,这些都是为了避免搜索过早陷入停止状态。该算法的主要思想是:一方面加强正反馈的效果,提高蚂蚁的搜索效率;另一方面,采取一定措施,减小陷入局部优化的可能性。改进后的算法流程如图 9-9 所示。

在具体介绍该算法之前,首先给出几个相关的定义。

定义 1:所有蚂蚁完成一次搜索,被称为一次周游。

定义 2:若干只蚂蚁各自进行一次搜索后,这些搜索结果中最好的一个即为周游最优路线。

定义 3:已经完成的所有搜索中,结果最好的行进路线即为全局最优路线。

在蚁群系统中,只更新构成全局最优路线的边上的信息素,而在 MMAS 中,提出了一种新的信息素更新策略,即更新周游最优路线。在搜索的初期,只更新周游最优路线,然后,逐渐提高全局最优路线的更新频率,直至只更新全局最优路线。实验证明,这种方法可在一定程度上改进搜索的结果。

MMAS 算法是在蚂蚁系统算法基础上进行了许多改进之后的算法,主要表现在以下三个方面:

(1) 在算法运行期间更多地利用最优解信息,即每次迭代后仅允许一只最优的蚂蚁增加信息素,该最优蚂蚁可以是当代最优的,也可以是全局最优的。

(2) 为了尽量避免搜索停止现象,本算法对信息素进行了限制,这也是将该算法称为最大-最小蚂蚁系统的原因。

(3) 在开始搜索前,将所有边的信息素水平设为信息素最大值,即本算法将信息素初始化为最大值,这样初始化有利于算法在最初阶段搜索到更多的解。这样在搜索的初期,蚂蚁的搜索范围较大,从而减少了搜索停滞于局部最优的情况发生。

改进后的算法如下:

```
    pls = 0.1;                          % 局部寻优阈值 pls(相当于变异率)
solution_temp = zeros(L,N+1);
  k = 1;
  while(k <= L)
        solution_temp(k,:) = solution_ascend(k,:);
        rp = rand(1,N);                 % 产生一个 1 * N(51)维的随机数组
        for i = 1:N
            if rp(i) <= pls             % 某值小于 pls 则随机改变其对应的路径标识
```

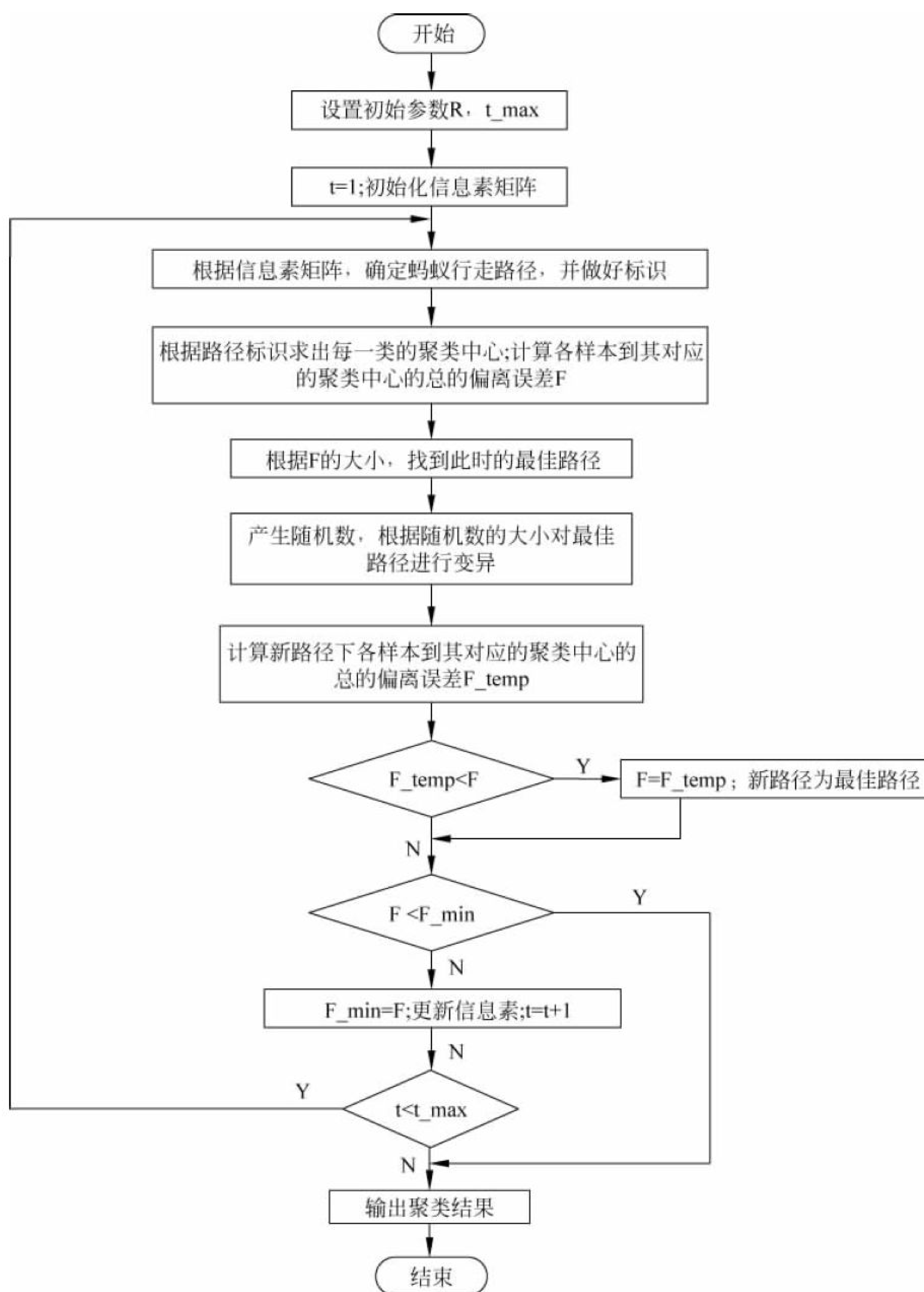


图 9-9 改进蚁群算法流程图

```
                current_cluster_number = setdiff([1:K],solution_temp(k,i));
                rrr = randint(1,1,[1,K-1]);
                change_cluster = current_cluster_number(rrr);
                solution_temp(k,i) = change_cluster;
            end
        end
```

9.4.2 完整程序及仿真结果

MMAS算法程序代码如下：

```
clc;
clf;
clear;
 %X = 测试样本矩阵
X = load('data1.txt');
[N,n] = size(X);              %N = 测试样本数;n = 测试样本的属性数
K = 4;                        %K = 组数
R = 100;                      %R = 蚂蚁数
t_max = 1000;                 %t_max = 最大迭代次数
%初始化
c = 10 ^ - 2;
tau = ones(N,K) * c;          %信息素矩阵,初始值为 0.01 的 N * K 矩阵(样本数 * 聚类数)
q = 0.9;                      %阈值 q
rho = 0.1;                    %蒸发率
best_solution_function_value = inf; %最佳路径度量值(初值为无穷大,该值越小聚类效果越好)
tic                           %计算程序运行时间
t = 1;
while ((t <= t_max))          %进行 t_max 次迭代计算
    %路径标识字符: 标识每只蚂蚁的路径
    solution_string = zeros(R,N + 1);
    for i = 1 : R             %以信息素为依据确定蚂蚁的路径
          r = rand(1,N);      %随机产生值为 0～1 随机数的 1 * 51 的数组
          for g = 1 : N
              if r(g) < q     %如果 r(g)小于阈值
                    tau_max = max(tau(g,:));
                    %聚类标识数,选择信息素最多的路径
                    Cluster_number = find(tau(g,:) == tau_max);
                      %确定第 i 只蚂蚁对第 g 个样本的路径标识
                    solution_string(i,g) = Cluster_number(1);
                %如果 r(g)大于阈值,求出各路径信息素占总信息素的比例,按概率选择路径
                else
                   sum_p = sum(tau(g,:));
                   p = tau(g,:) / sum_p;
                   for u = 2 : K
                       p(u) = p(u) + p(u - 1);
                   end
                  rr = rand;
                   for s = 1 : K
                        if (rr <= p(s))
                             Cluster_number = s;
                             solution_string(i,g) = Cluster_number;
                        break;
                   end
              end
```

```
            end
        end
        % 计算聚类中心
        weight = zeros(N,K);
           for h = 1:N                                       % 给路径做计算标识
                 Cluster_index = solution_string(i,h);       % 类的索引编号
                 weight(h,Cluster_index) = 1;     % 对样本选择的类在 weight 数组的相应位置标 1
           end
           cluster_center = zeros(K,n);                      % 聚类中心(聚类数 K 个中心)
           for j = 1:K
                 for v = 1:n
                       sum_wx = sum(weight(:,j).*X(:,v));  % 各类样本各属性值之和
                       sum_w = sum(weight(:,j));             % 各类样本个数
                       if sum_w == 0                       % 该类样本数为 0,则该类的聚类中心为 0
                          cluster_center(j,v) = 0
                           continue;
                       else            % 该类样本数不为 0,则聚类中心的值取样本属性值的平均值
                       cluster_center(j,v) = sum_wx/sum_w;
                       end
                    end
                end
    % 计算各样本点各属性到其对应的聚类中心的均方差之和,该值存入 solution_string 的最后一位
                F = 0;
                for j = 1:K
                      for ii = 1:N
                           Temp = 0;
                           if solution_string(i,ii) == j;
                                for v = 1:n
                                     Temp = ((abs(X(ii,v) - cluster_center(j,v))).^2) + Temp;
                                end
                                Temp = sqrt(Temp);
                      end
                    F = (Temp) + F;
                  end
              end
               solution_string(i,end) = F;
        end
        % 根据 F 值,把 solution_string 矩阵升序排序
        [fitness_ascend,solution_index] = sort(solution_string(:,end),1);
        solution_ascend = [solution_string(solution_index,1:end-1) fitness_ascend];
        pls = 0.1;                         % 局部寻优阈值 pls(相当于变异率)
        L = 2;                             % 在 L 条路径内局部寻优
        % 局部寻优程序
        solution_temp = zeros(L,N+1);
        k = 1;
        while(k <= L)
                solution_temp(k,:) = solution_ascend(k,:);
                rp = rand(1,N);        % 产生一个 1*N(51)维的随机数组,某值小于 pls 则随机改变
                                       % 其对应的路径标识
                for i = 1:N
                    if rp(i) <= pls
                          current_cluster_number = setdiff([1:K],solution_temp(k,i));
```

```
                    rrr = randint(1,1,[1,K - 1]);
                    change_cluster = current_cluster_number(rrr);
                    solution_temp(k,i) = change_cluster;
                end
            end
        %计算临时聚类中心
         solution_temp_weight = zeros(N,K);
         for h = 1:N
             solution_temp_cluster_index = solution_temp(k,h);
             solution_temp_weight(h,solution_temp_cluster_index) = 1;
         end
         solution_temp_cluster_center = zeros(K,n);
         for j = 1:K
              for v = 1:n
                     solution_temp_sum_wx = sum(solution_temp_weight(:,j).*X(:,v));
                     solution_temp_sum_w = sum(solution_temp_weight(:,j));
                     if solution_temp_sum_w == 0
                     solution_temp_cluster_center(j,v) = 0;
                     continue;
                     else
                            solution_temp_cluster_center(j,v) = solution_temp_sum_wx/
solution_temp_sum_w;
                     end
                 end
         end
          %计算各样本点各属性到其对应的临时聚类中心的均方差之和 Ft
         solution_temp_F = 0;
         for j = 1:K
              for ii = 1:N
                     st_Temp = 0;
                     if solution_temp(k,ii) == j;
                            for v = 1:n
                               st_Temp = ((abs(X(ii,v) - solution_temp_cluster_center(j,
v))).^2) + st_Temp;
                            end
                            st_Temp = sqrt(st_Temp);
                     end
                     solution_temp_F = (st_Temp) + solution_temp_F;
              end
         end
        solution_temp(k,end) = solution_temp_F;
        %根据临时聚类度量调整路径
        %如果 Ft<Fl,则 Fl = Ft,Sl = St
         if solution_temp(k,end) <= solution_ascend(k,end)
              solution_ascend(k,:) = solution_temp(k,:);
         end
         if solution_ascend(k,end)<= best_solution_function_value
               best_solution = solution_ascend(k,:);
         end
    k = k + 1;
```

```
        end
       % 用最好的 L 条路径更新信息素矩阵
      tau_F = 0;
      for j = 1:L
          tau_F = tau_F + solution_ascend(j,end);
      end
      for i = 1 : N
          tau(i,best_solution(1,i)) = (1 - rho) * tau(i,best_solution(1,i)) + 1/ tau_F;
       % 1/tau_F 和 rho/tau_F 效果都很好
      end
      t = t + 1
      best_solution_function_value = solution_ascend(1,end);
      best_solution_function_value
  end
  time = toc;                        % 输出程序运行时间
  clc
  t
  time
  cluster_center
  best_solution = solution_ascend(1,1:end - 1);
  IDY = ctranspose(best_solution)
  best_solution_function_value = solution_ascend(1,end)
   % 分类结果显示
  plot3(cluster_center(:,1),cluster_center(:,2),cluster_center(:,3),'o');grid;box
  title('蚁群聚类结果(R = 100,t = 10000)')
  xlabel('X')
  ylabel('Y')
  zlabel('Z')
  YY = [1 2 3 4];
  index1 = find(YY(1) == best_solution)
  index2 = find(YY(2) == best_solution)
  index3 = find(YY(3) == best_solution)
  index4 = find(YY(4) == best_solution)
  line(X(index1,1),X(index1,2),X(index1,3),'linestyle','none','marker',' * ','color','g');
  line(X(index2,1),X(index2,2),X(index2,3),'linestyle','none','marker',' * ','color','r');
  line(X(index3,1),X(index3,2),X(index3,3),'linestyle','none','marker',' + ','color','b');
  line(X(index4,1),X(index4,2),X(index4,3),'linestyle','none','marker','s','color','b');
  rotate3d
```

程序运行完后,仿真结果如图 9-10 所示。从图中可以看出 MMAS 聚类效果比基本蚁群聚类效果要好,但分类效果还不是太好,说明该三元色不适合使用该算法分类。

程序运行结果如下:

```
t =
1001
time =
84.9270
cluster_center =
```

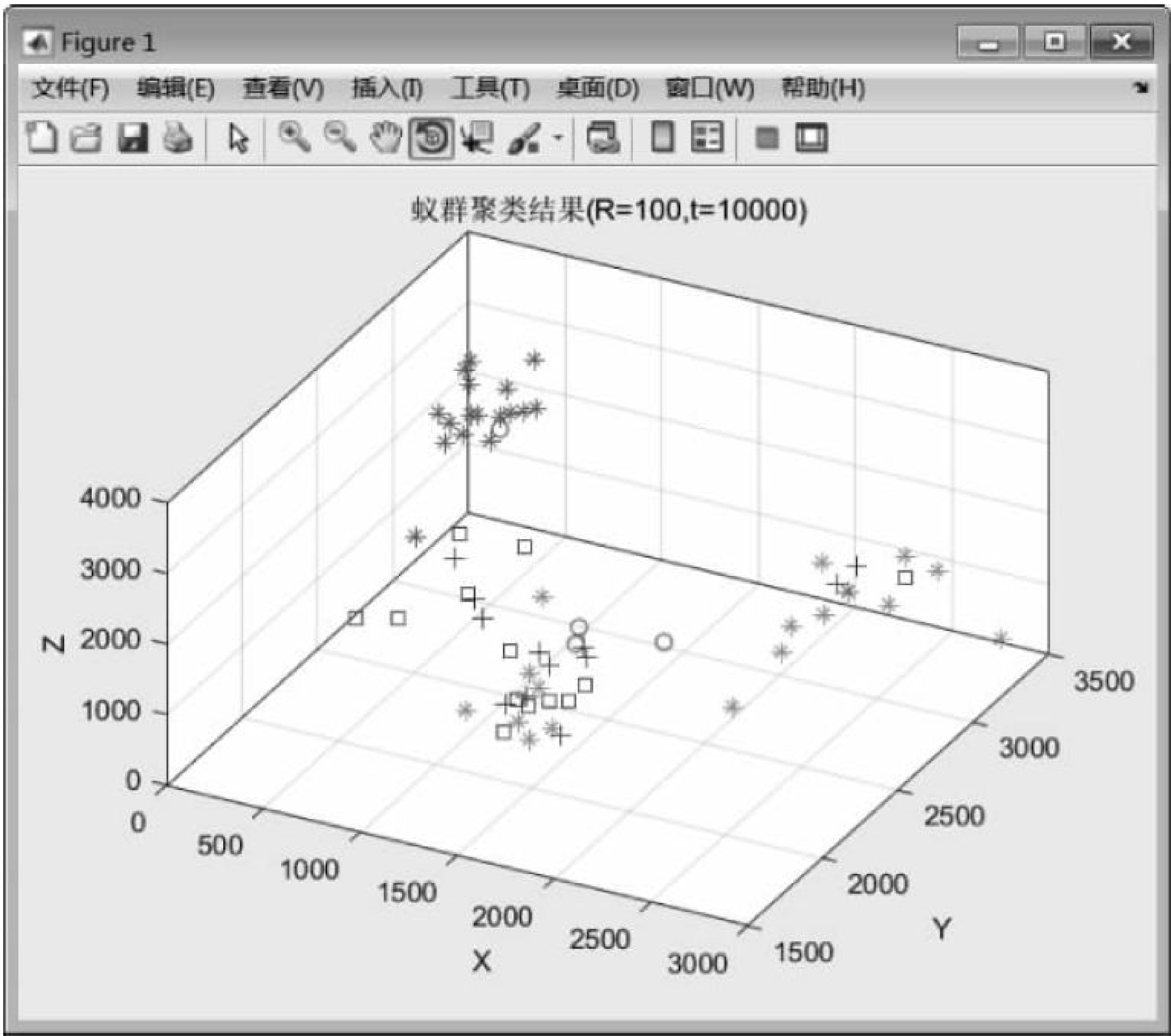


图 9-10 MMAS 聚类结果

```
1.0e+03 *
     1.9095     2.3453     1.6705
     0.4709     3.1052     2.2664
     1.7053     2.0221     2.1305
     1.6203     2.1557     2.0522
best_solution_function_value =
    4.1595e+04
index1 =
1～15 列
1    3    8    14    15    19    22    24    26    33    36    39    41    43
45
16 列
47
index2 =
1～15 列
2    5    6    9    10    12    13    23    27    28    29    34    38    44
46
16～18 列
48    49    55
index3 =
11    16    17    18    20    37    40    42    50    52    56    58
index4 =
7    21    25    30    31    32    35    51    53    54    57    59
```

9.5 结论

改进基本蚁群算法之后缩短了迭代次数，减少了计算量，聚类效果要好于基本蚁群算法。但是，从整体上来说两种算法的聚类效果都不太好，说明该算法不适合于酒瓶的分类，体现了蚁群算法的局限性。蚁群算法虽被成功应用到了旅行商问题上，但大家以后在应用该算法时，还是应该根据具体的问题来定。

习题

（1）简述蚁群算法的基本原理。
（2）简述蚁群算法的特点。

第10章

粒子群算法聚类设计

粒子群算法(particle swarm optimization,PSO)是1995年由美国社会心理学家James Kennedy和电气工程师Russell Eberhart在鸟群、鱼群和人类社会的行为规律的启发下提出的一种基于群智能的演化计算技术。由于算法收敛速度快,需要设置、调整的参数少,实现简洁,近年来受到学术界的广泛重视。

10.1 粒子群算法简介

粒子群算法(PSO)是继蚁群算法之后的又一种新的群体智能算法,目前已成为进化算法的一个重要分支。粒子群算法的基本思想是模拟鸟类群体行为,并利用了生物学的生物群体模型。该模型中描述了鸟类生活中使用了简单的规则来确定自己的飞行方向和飞行速度,并且成功地寻找到栖息地。这些规则表述为:①飞离最近的个体;②飞向目标;③飞向群体的中心。Heppner受鸟类的群体智能启发,建立了该模型。Eberhart和Kennedy对Heppner的模型进行了修正,同时引入了人类的个体学习和整体文化形成的模式,一方面个体向周围的优秀者的行为学习,另一方面个体不断总结自己的经验形成自己的知识库,从而提出了粒子群算法。该算法由于运算速度快、局部搜索能力强、参数设置简单,受学术界的广泛重视,现在粒子群算法在函数优化、神经网络训练、模式分类、模糊系统控制以及其他工程领域都得到了广泛应用。

10.2 经典的粒子群算法的运算过程

经典粒子群算法和其他的进化算法相似,也采用"群体"与"进化"的概念,同样是根据个体即微粒(particle)的适应度大小进行操作。所不同的是,微粒群算法不像其他进化算法那样对个体使用进化算子,而是将每个个体看作是在N维搜索空间中的一个无重量无体积的微粒,并在搜索空间中以一定的速度飞行。该飞行速度根据个体的飞行经验和群体的飞行

经验来进行动态调整。

Kennedy 和 Eberhart 最早提出的 PSO 算法进化方程为

$$v_{ij}(t+1)=v_{ij}(t)+c_1\cdot r_1\cdot(p_{ij}(t)-x_{ij}(t))+c_2\cdot r_2\cdot(p_{gj}(t)-x_{ij}(t)) \quad (10\text{-}1)$$

$$x_{ij}(t+1)=x_{ij}(t)+v_{ij}(t+1) \quad (10\text{-}2)$$

其中，i 表示第 i 个微粒，j 表示的是微粒 i 的第 j 维分量，t 表示第 t 代，学习因子 c_1 和 c_2 为非负常数，c_1 用来调节微粒向本身最好位置飞行的步长，c_2 用来调节微粒向群体最好位置飞行的步长，通常 c_1 和 c_2 在[0,2]取值。

迭代终止条件根据具体问题一般选为最大迭代次数或粒子群搜索到的最优位置满足于预先设定的精度。

经典微粒群算法的流程如下：

(1) 依照如下步骤初始化，对微粒群的随机位置和速度进行初始设定。

① 设定群体规模，即粒子数为 N。

② 对任意 i,j，随机产生 x_{ij}，v_{ij}。

③ 对任意 i 初始化局部最优位置为 $p_i=x_i$。

④ 初始化全局最优位置 p_g。

(2) 根据目标函数，计算每个微粒的适应度值。

(3) 对于每个微粒，将其适应度值与其本身所经历过的最好位置 p_i 的适应度值进行比较，如更好，则将现在的 x_i 位置作为新的 p_i。

(4) 对每个微粒，将其经过的最好位置的 p_i 适应度值与群体的最好位置的适应度值比较，如果更好，则将 p_i 的位置作为新的 p_g。

(5) 对微粒的速度和位置进行更替。

如未达到终止条件，则返回(2)。

10.3 两种基本的进化模型

Kennedy 等在对鸟群觅食的观察过程中发现，每只鸟并不总是能看到鸟群中其他所有鸟的位置和运动方向，而往往只是看到相邻的鸟的位置和运动方向。因此提出了两种粒子群算法模型：全局模型(global version PSO)和局部模型(local version PSO)。

在基本的 PSO 算法中，根据直接相互作用的微粒群定义可构造 PSO 算法的两种不同版本，也就是说，可以通过定义全局最好微粒(位置)或局部最好微粒(位置)构造具有不同行为的 PSO 算法。

1. G_{best}模型(全局最好模型)

G_{best}模型以牺牲算法的健壮性为代价提高算法的收敛速度，基本 PSO 算法就是该模型的典型体现。在该模型中，整个算法以该微粒(全局最好的微粒)为吸引子，将所有微粒拉向它，使所有的微粒最终收敛于该位置。如果在进化过程中，该全局最优解得不到更新，则微粒群将出现类似于遗传算法早熟的现象。

2. L_{best}模型（局部最好模型）

为了防止 G_{best}模型可能出现早熟现象，L_{best}模型采用多个吸引子代替 G_{best}模型中的单一吸引子。首先将为粒子群分解为若干个子群，在每个粒子群中保留其局部最好微粒 $p_i(t)$，称为局部最好的位置或邻域最好位置。

实验表明，局部最好模型的 PSO 比全局最好模型的收敛慢，但不容易陷入局部最优解。

10.4 改进的粒子群优化算法

10.4.1 粒子群优化算法原理

最初的 PSO 是从解决连续优化问题发展起来的，Eberhart 等又提出了 PSO 的二进制版本，来解决工程实际中的优化问题。

粒子群算法是一种局部搜索效率高的搜索算法，收敛快，特别是在算法的早期，但也存在着精度较低、易发散等缺点。若加速系数、最大速度等参数太大，粒子群可能错过最优解，算法不能收敛；而在收敛的情况下，由于所有的粒子都同时向最优解的方向飞去，所以粒子趋向同一化（失去了多样性），这样就使得算法容易陷入局部最优解，即算法收敛到一定精度时，无法继续优化，因此很多学者都致力于提高 PSO 算法的性能。

Y. Shi 和 Eberhart 在 1998 年对微粒群算法引入了惯性权重 $w(t)$，并提出了在进化过程中线性调整惯性权重的方法，来平衡全局和局部搜索的性能，该方程已被学者们称为标准 PSO 算法。

粒子群优化算法具有进化计算和群智能特点。与其他进化算法相类似，粒子群算法也是通过个体间的协作与竞争，实现复杂空间中最优解的搜索。

粒子群优化算法中，每一个优化问题的解可看做是搜索空间中的一只鸟，即“粒子”。首先生成初始种群，即在可行解空间中随机初始化一群粒子，每个粒子都为优化问题的一个可行解，并由目标函数为之确定一个适应度值。每个粒子都将在解空间中运动，并由运动速度决定其飞行方向和距离。通常粒子将追随当前的最优粒子在解空间中搜索。在每一次迭代中，粒子将跟踪两个“极值”来更新自己，一个是粒子本身找到的最优解，另一个是整个种群目前找到的最优解，这个极值即全局极值。

粒子群算法可描述为：设粒子群在一个 n 维空间中搜索，由 m 个粒子组成种群 $Z=\{Z_1,Z_2,\cdots,Z_m\}$，其中的每个粒子所处的位置 $Z_i=\{z_{i1},z_{i2},\cdots,z_{in}\}$都表示问题的一个解，粒子通过不断调整自己的位置 Z_i 来搜索新解。每个粒子都能记住自己搜索到的最好解，记作 p_{id}，以及整个粒子群经历过的最好的位置，即目前搜索到的最优解，记作 p_{gd}。此外每个粒子都有一个速度，记作 $V_i=\{v_{i1},v_{i2},\cdots,v_{in}\}$。当两个最优解都找到后，每个粒子根据式(10-3)来更新自己的速度。

$$v_{id}(t+1)=wv_{id}(t)+\eta_1 r_1(p_{id}-z_{id}(t))+\eta_2 r_2(p_{gd}-z_{id}(t)) \tag{10-3}$$

$$z_{id}(t+1)=z_{id}(t)+v_{id}(t+1) \tag{10-4}$$

式中，$v_{id}(t+1)$表示第 i 个粒子在 $t+1$ 次迭代中第 d 维上的速度，w 为惯性权重，η_1、η_2 为加速常数，r_1、r_2 为 0～1 的随机数。此外，为使粒子速度不致过大，可以设置速度上限 v_{max}，当式(10-1)中 $v_{id}(t+1)>v_{max}$时，$v_{id}(t+1)=v_{max}$；$v_{id}(t+1)<-v_{max}$时，$v_{id}(t+1)=-v_{max}$。

从式(10-3)和式(10-4)可以看出，粒子的移动方向由三部分决定：自己原有的速度 $v_{id}(t)$、与自己最佳经历的距离 $p_{id}-z_{id}(t)$、与群体最佳经历的距离 $p_{gd}-z_{id}(t)$，并分别由权重系数 w、η_1、η_2 决定其重要性。

下面介绍这些参数的设置。PSO 算法中需要调节的参数主要包括：

1）加速度因子 η_1、η_2，即学习因子

学习因子(也称加速度系数)η_1 和 η_2 分别调节粒子向全局最优粒子和个体最优粒子方向飞行的最大步长。若太小，则粒子可能远离目标区域；若太大则可能导致粒子忽然向目标区域飞去或飞过目标区域。合适的 η_1 和 η_2 可以加快收敛且不易陷入局部最优，目前大多数文献均采用 $\eta_1=\eta_2=2$。

2）种群规模 N

PSO 算法种群规模较小，一般令 $N=20\sim40$。其实对于大部分问题取 10 个粒子就能得到很好的结果，但对于较难或者特定类别的问题，粒子数可能取到 100 或 200。

3）适应度函数

$$F=\sum_{j=1}^{k}\sum_{i=1}^{s}\boldsymbol{w}_{ij}\sum_{p=1}^{n}(X_{ip}-C_{jp})^2 \tag{10-5}$$

其中，w_{ij}是 0，1 矩阵，当 x 属于该类时元素为 0，否则为 1。

4）惯性权重系数 w

$$w=w_{max}-t\times\frac{w_{max}-w_{min}}{t_{max}} \tag{10-6}$$

惯性权重系数 w 用来控制前面的速度对当前速度的影响，较大的 w 可以加强 PSO 的全局搜索能力，而较小的能加强局部搜索能力。目前普遍采用将 w 设置为从 0.9 到 0.1 线性下降的方法，这种方法可使得 PSO 在开始时探索较大的区域，较快地定位最优解的大致位置，随着 w 逐渐减小，粒子速度减慢，开始精细局部搜索。

10.4.2 粒子群优化算法的基本流程

粒子群算法的基本流程如图 10-1 所示。

粒子群算法的基本步骤如下：

(1) 初始化粒子群，即随机设定各粒子的初始位置和初始速度 V。

(2) 根据初始位置和速度产生各粒子新的位置。

(3) 计算每个粒子的适应度值。

(4) 对于每个粒子，比较它的适应度值和它经历过的最优位置 p_{id}的适应度值，如果更好则更新。

(5) 对于每个粒子，比较它的适应度值和群体所经历的最优位置 p_{gd}的适应度值，如果更好则更新 p_{gd}。

(6) 根据式(10-3)和式(10-4)调整粒子的速度和位置。

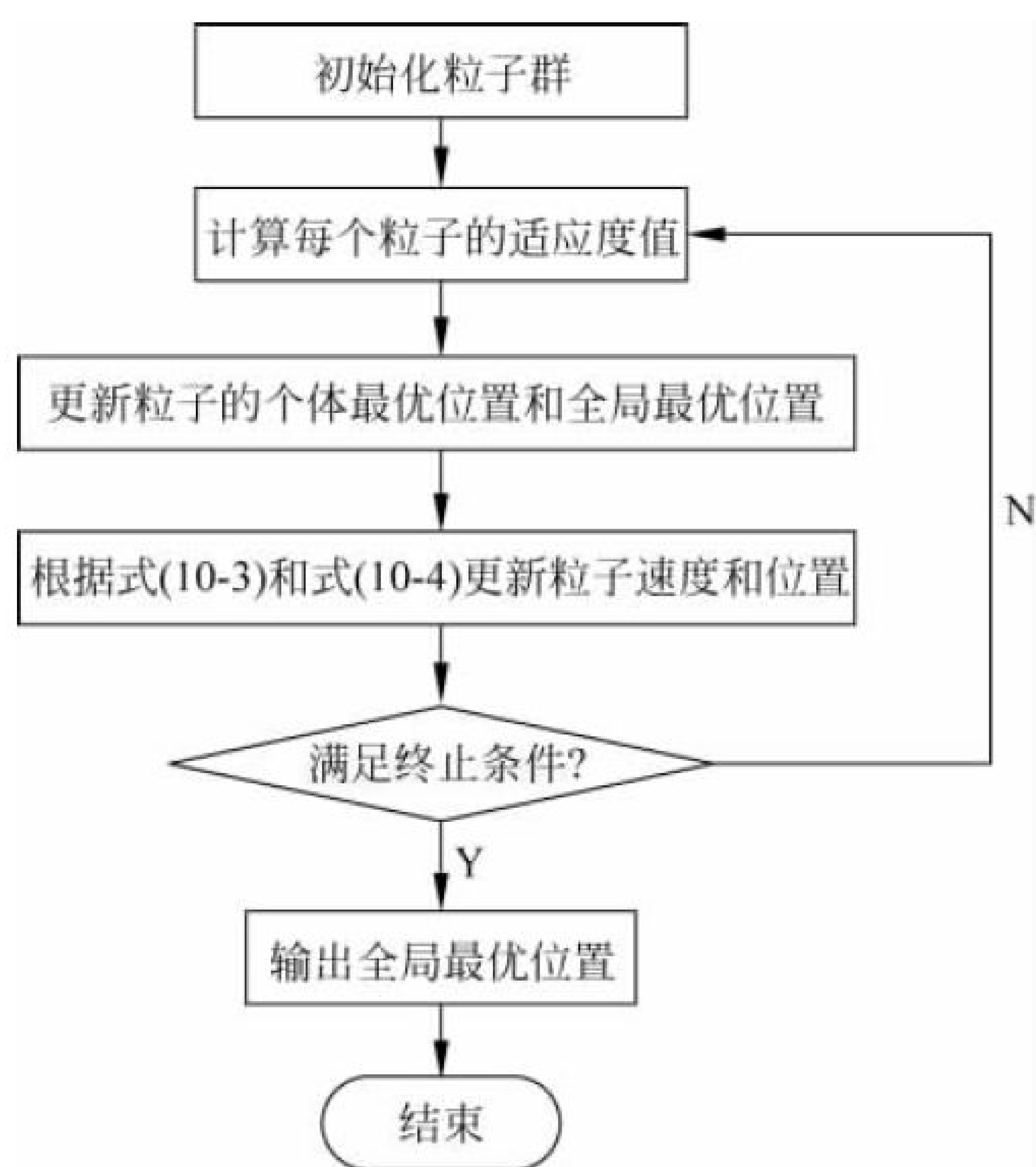


图 10-1 粒子群算法流程图

（7）如果达到终止条件（足够好的位置或最大迭代次数），则结束，否则转步骤（3）继续迭代。

10.5 粒子群算法与其他算法的比较

1. 相同点

粒子群算法与其他进化算法（如遗传算法和蚁群算法）有许多相似之处：

（1）粒子群算法和其他进化算法都基于“种群”概念，用于表示一组解空间中的个体集合。它们都随机初始化种群，使用适应度值来评价个体，而且都根据适应度值来进行一定的随机搜索，并且不能保证一定能找到最优解。

（2）种群进化过程采用子代与父代竞争机制，若子代具有更好的适应度值，则子代将替换父代，因此都具有一定的选择性。

（3）算法都具有并行性，即搜索过程是从一个解集合开始的，而不是从单个个体开始的，不容易陷入局部极小值。并且这种并行性易于在并行计算机上实现，从而提高算法的性能和效率。

2. 不同点

粒子群算法与其他进化算法的区别：

（1）粒子群算法在进化过程中同时记忆位置和速度信息，而遗传算法和蚁群算法通常只记忆位置信息。

（2）粒子群算法的信息通信机制与其他进化算法不同。遗传算法中染色体间通过交叉等操作进行通信，蚁群算法中每只蚂蚁以蚁群全体构成的信息素轨迹作为通信机制，因此整

个种群比较均匀地向最优区域移动。在全局模式的粒子群算法中,只有全局最优粒子提供信息给其他的粒子,整个搜索更新过程是跟随当前最优解的过程,因此所有的粒子很可能更快地收敛于最优解。

10.6 粒子群算法分类器的 MATLAB 实现

本例采用表 1-2 中的三元色数据,按照颜色数据所表征的特点,将数据按照各自所属的类别归类。

由于粒子群优化算法是迭代求取最优值,所以事先无须训练数据,故取后 30 组数据确定类别。下面使用 MATLAB 构建粒子群优化算法。

10.6.1 设定参数

PSO 优化算法需要设定粒子的学习因子(速度更新参数)、最大迭代次数以及惯性权重初始和终止值及聚类类数。

参数设定程序代码如下:

```
c1 = 1.6;c2 = 1.6;                 %设定学习因子值(速度更新参数)
wmax = 0.9;wmin = 0.4;             %设定惯性权重初始及终止值
M = 1800;                          %最大迭代数
K = 4;                             %类别数
```

10.6.2 初始化

算法还需将粒子的位置、速度和其他一些变量进行初始化。

初始化程序代码如下:

```
fitt = inf * ones(1,N);            %初始化个体最优适应度
fg = inf;                          %初始化群体最优适应度
fljg = clmat(1,:);                 %当前最优分类
v = rand(N,K * D);                 %初始速度
x = zeros(N,K * D);                %初始化粒子群位置
y = x;                             %初始化个体最优解
pg = x(1,:);                       %初始化群体最优解
cen = zeros(K,D);                  %类别中心定维
fitt2 = fitt;                      %粒子适应度定维
```

10.6.3 完整程序及仿真结果

粒子群优化算法的完整 MATLAB 程序代码如下:

```
clc;
clear all;
format long;
tic
data = [1702.8    1639.79    2068.74
        1877.93   1860.96    1975.3
        867.81    2334.68    2535.1
        1831.49   1713.11    1604.68
        460.69    3274.77    2172.99
        2374.98   3346.98    975.31
        2271.89   3482.97    946.7
        1783.64   1597.99    2261.31
        198.83    3250.45    2445.08
        1494.63   2072.59    2550.51
        1597.03   1921.52    2126.76
        1598.93   1921.08    1623.33
        1243.13   1814.07    3441.07
        2336.31   2640.26    1599.63
        354       3300.12    2373.61
        2144.47   2501.62    591.51
        426.31    3105.29    2057.8
        1507.13   1556.89    1954.51
        343.07    3271.72    2036.94
        2201.94   3196.22    935.53
        2232.43   3077.87    1298.87
        1580.1    1752.07    2463.04
        1962.4    1594.97    1835.95
        1495.18   1957.44    3498.02
        1125.17   1594.39    2937.73
        24.22     3447.31    2145.01
        1269.07   1910.72    2701.97
        1802.07   1725.81    1966.35
        1817.36   1926.4     2328.79
        1860.45   1782.88    1875.13];
% -------- 参数设定 -----------
N = 70;                                   % 粒子数
c1 = 1.6;c2 = 1.6;                        % 设定学习因子值(速度更新参数)
wmax = 0.9;wmin = 0.4;                    % 设定惯性权重初始及终止值
M = 1600;                                 % 最大迭代数
K = 4;                                    % 类别数
[S D] = size(data);                       % 样本数和特征维数
% -------- 初始化 ----------------
for i = 1:N
clmat(i,:) = randperm(S);                 % 随机取整数
end
clmat(clmat > K) = fix(rand * K + 1);     % 取整函数
fitt = inf * ones(1,N);                   % 初始化个体最优适应度
fg = inf;                                 % 初始化群体最优适应度
fljg = clmat(1,:);                        % 当前最优分类
v = rand(N,K * D);                        % 初始速度
```

```
x = zeros(N,K * D);                          % 初始化粒子群位置
y = x;                                       % 初始化个体最优解
pg = x(1,:);                                 % 初始化群体最优解
cen = zeros(K,D);                            % 类别中心定维
fitt2 = fitt;                                % 粒子适应度定维
% ------ 循环优化开始 ------------
for t = 1:M
for i = 1:N
    ww = zeros(S,K);                         %% 产生零矩阵
    for ii = 1:S
         ww(ii,clmat(i,ii)) = 1;             % 加权矩阵,元素非 0 即 1
    end
    ccc = [];tmp = 0;
    for j = 1:K
          sumcs = sum(ww(:,j) * ones(1,D). * data);
          countcs = sum(ww(:,j));
         if countcs == 0
            cen(j,:) = zeros(1,D);
         else
            cen(j,:) = sumcs/countcs;        % 求类别中心
         end
         ccc = [ccc,cen(j,:)];               % 串联聚类中心
         aa = find(ww(:,j) == 1);
         if length(aa)~ = 0
              for k = 1:length(aa)
                tmp = tmp + (sum((data(aa(k),:) - cen(j,:)).^2));   %% 适应度计算
              end
         end
    end
    x(i,:) = ccc;
    fitt2(i) = tmp;                          %% 适应度值 Fitness value
end
% 更新群体和个体最优解
for i = 1:N
         if fitt2(i)<fitt(i)
              fitt(i) = fitt2(i);
              y(i,:) = x(i,:);               % 个体最优
              if fitt2(i)<fg
              pg = x(i,:);                   % 群体最优
              fg = fitt2(i);                 % 群体最优适应度
              fljg = clmat(i,:);             % 当前最优聚类
              end
         end
    end
bfit(t) = fg;                                % 最优适应度记录
w = wmax - t * (wmax - wmin)/M;              % 更新权重,线性递减权重法的粒子群算法
     for i = 1:N
          % 更新粒子速度和位置
```

```
            v(i,:) = w * v(i,:) + c1 * rand(1,K * D). * (y(i,:) - x(i,:)) + c2 * rand(1,K * D). * (pg - x(i,:));
                x(i,:) = x(i,:) + v(i,:);
                for k = 1:K
                cen(k,:) = x((k - 1) * D + 1:k * D);            % 拆分粒子位置,获得 K 个中心
                end
                % 重新归类
                for j = 1:S
                          tmp1 = zeros(1,K);
                          for k = 1:K
                          tmp1(k) = sum((data(j,:) - cen(k,:)).^2);  % 每个样本关于各类的距离
                          end
                          [tmp2 clmat(i,j)] = min(tmp1);    % 最近距离归类
                end
        end
end
% ------ 循环结束 ------------
M                                                               % 迭代次数
fljg                                                            % 最优聚类输出
fg                                                              % 最优适应度输出
figure(1)
plot(bfit);                                                     % 绘制最优适应度轨迹
xlabel('种群迭代次数');
ylabel('适应度');
title('适应度曲线');
cen                                                             % 聚类中心
toc
```

仿真适应度曲线如图 10-2 所示。

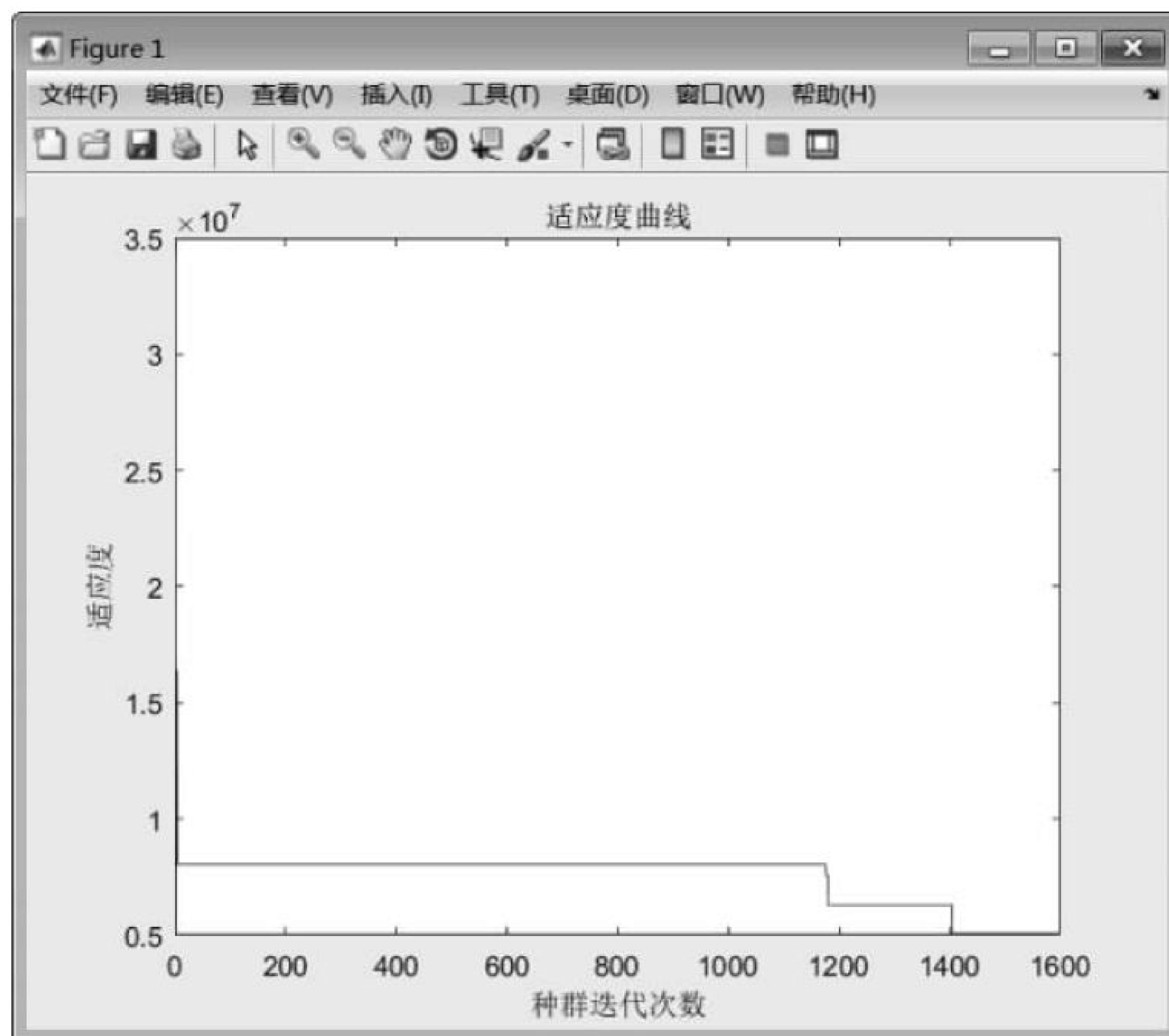


图 10-2　PSO 算法适应度曲线

PSO算法仿真适应度准确值为：适应度＝5.07E＋06。

对预测样本值的仿真输出结果如下：

```
Fljg(最优聚类输出) =
   1 ～ 16 列
   2     2     4     2     4     3     3     2     4     2     2     2     1
   3     4     3
   17 ～ 30 列
   4     2     4     3     3     2     2     1     1     4     1     2     2     2
Fg(最优适应度值) =
5.074427169449084e+06
Cen(数据聚类中心) =
   1.0e+03 *
   1.583318501485389   1.635223368390102   2.714581718419658
   2.181834667703786   1.404155371279036   2.423318117482111
   2.998901968724054   2.358864196437189   2.063591486634854
   3.502722145507502   2.296065717941481   1.666953189720583
```

调整显示方式后，PSO算法聚类结果与标准结果对比如表10-1所示。

表10-1　PSO聚类结果与标准结果对比

A	B	C	标准类别	PSO分类
1702.8	1639.79	2068.74	3	3
1877.93	1860.96	1975.3	3	3
867.81	2334.68	2535.1	4	4
1831.49	1713.11	1604.68	3	3
460.69	3274.77	2172.99	2	2
2374.98	3346.98	975.31	1	1
2271.89	3482.97	946.7	1	1
1783.64	1597.99	2261.31	3	3
198.83	3250.45	2445.08	2	2
1494.63	2072.59	2550.51	4	4
1597.03	1921.52	2126.76	3	3
1598.93	1921.08	1623.33	3	3
1243.13	1814.07	3441.07	4	4
2336.31	2640.26	1599.63	1	1
354	3300.12	2373.61	2	2
2144.47	2501.62	591.51	1	1
426.31	3105.29	2057.8	2	2
1507.13	1556.89	1954.51	3	3
343.07	3271.72	2036.94	2	2
2201.94	3196.22	935.53	1	1
2232.43	3077.87	1298.87	1	1
1580.1	1752.07	2463.04	3	3
1962.4	1594.97	1835.95	3	3

续表

A	B	C	标准类别	PSO 分类
1495.18	1957.44	3498.02	4	4
1125.17	1594.39	2937.73	4	4
24.22	3447.31	2145.01	2	2
1269.07	1910.72	2701.97	4	4
1802.07	1725.81	1966.35	3	3
1817.36	1927.4	2328.79	3	3
1860.45	1782.88	1875.13	3	3

10.7 结论

通过粒子群算法能够很快实现分类，而且通过惯性权重系数的线性更新，可以防止局部最优输出。虽然运行时间稍有增加，但效果明显。每种聚类数目下的最优聚类可以根据输出的适应度 fg 来判断，适应度值越小越好，并且需多次运行判断。

习题

(1) 什么是粒子群算法？
(2) 简述粒子群优化算法的原理。
(3) 简述粒子群算法流程。

参 考 文 献

[1] 周润景,张丽娜.基于 MATLAB 与 fuzzyTECH 的模糊与神经网络设计[M]. 北京：电子工业出版社,2010.

[2] 杨淑莹.模式识别与智能计算：MATLAB技术实现[M].北京：电子工业出版社,2008.

[3] 王小川,史峰,郁磊,等. MATLAB 神经网络 43 个案例分析[M].北京：北京航空航天大学出版社,2013.

[4] 田景文,高美娟.人工神经网络算法研究及应用[M]. 北京：北京理工大学出版社,2006.